International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 2
DOI: https://doi.org/10.17762/ijritcc.v11i2.6134
***Article Received**: 26 December 2022 **Revised**: 16 January 2023 **Accepted**: 28 January 2023*

A Hyper-parameter Tuning based Novel Model for Prediction of Software Maintainability

Rohit Yadav[1,*], Raghuraj Singh[2]
[1]Research Scholor, Department of Computer Science and Engineering
Dr. A.P.J. Abdul Kalam Technical University
Lucknow, India
[2]Professor, Department of Computer Science and Engineering
Harcourt Butler Technical University
Kanpur, India
Corresponding Author*: Rohit Yadav (e-mail: rohitatknit@gmail.com)

Abstract—Software maintainability is regarded as one of the most important characteristics of any software system. In today's digital world, the expanding significance of software maintenance is motivating the development of efficient software maintainability prediction (SMP) models using statistical and machine learning methods. This study proposes a hyper-parameter optimizable Software Maintainability Prediction (HPOSMP) model using the hybridized approach of data balancing and hyper-parameter optimization of Machine Learning (ML) approach using software maintainability datasets. The training dataset has been created with object-oriented software namely UIMS and QUES. To balance the dataset, Synthetic Minority Oversampling Technique (SMOTE) technology has been adopted. Further, Decision Tree, Gaussian Naïve Bayes, K-Nearest neighbour, Logistic Regression, and Support Vector Machine are adopted as Machine Learning and Statistical Regression Techniques for training of software maintainability dataset. Results demonstrate that the proposed HPOSMP model gives better performance as compared to the base SMP models.

Keywords- Software Quality; Software Maintenance; Machine Learning; Classification; Object oriented metric; Data balancing; Hyper-parameter Tuning; Accuracy;

I. INTRODUCTION

Software systems are becoming increasingly complex with time, and thus, software maintenance is gaining more attention and importance to fulfil the increased need of high-quality software in the software business [1]. Software Maintainability Prediction (SMP) gauges ease of carrying out various software maintenance tasks including adding or removing code or modifying already-existing code [2]. The maintenance phase accounts for nearly 60 to 70 percent of the overall cost of the software development life cycle (SDLC) and therefore, SMP is needed to reduce this cost [3] - [4]. Many SMP models have been developed by the researchers using statistical and Machine Learning (ML) approaches [5] - [7].

The potential issue with SMP is producing bias in the prediction models due to some infrequent and frequent extreme values of the instances. Also, like other machine learning training data, maintainability prediction dataset suffers from data imbalance. For example, the categorical variable can be treated as Low or High. Where Low refers to those object-oriented classes which require more efforts in maintenance phase and High refers to those object-oriented classes which require lesser efforts during maintenance phase. Thus, prior prediction of Low maintainable class aid to researchers or practitioners in putting optimal effort to those classes with Low maintenance and hence reducing the overall maintenance cost.

Prior to training for software maintainability, hyper-parameter tuning of statistical and machine learning approaches is recommended in order to increase the accuracy of the software maintainability prediction model [8]. In machine learning there are two types of parameters; first the model parameters which can be initialized and changed through the data learning process such as neurons in neural networks whereas hyper-parameters are the parameters that can be changed or tuned before the training process [9]. The variables known as hyper-parameters are parameters of machine learning techniques that can be tuned to calibrate more accurate software maintainability prediction models. This is also termed as Hyper-parameter optimization (HPO) [10].

Consequently, the novelty of our current investigation in comparison to earlier studies is as follows:

- Designing of Software maintainability prediction models using outliers' detection and outliers' replacements scheme.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 2
DOI: https://doi.org/10.17762/ijritcc.v11i2.6134
***Article Received**: 26 December 2022 **Revised**: 16 January 2023 **Accepted**: 28 January 2023*

- Forming categorical class variable of software maintainability parameter through the mean approach using the Change metric.
- Optimizing the SMP model by selecting the best combination of optimizable hyper-parameters of classification models.

On the basis of above proposed work, following research questions have been formulated and answered.

RQ.1: What effect does data balancing have on the distribution of classes in the software maintainability datasets?

RQ.2: How effective are the SMP models created using HPO in comparison to the models created using basic machine learning and logistic regression?

RQ.3: What is the Precision, Recall, and F1-Score ratings for each SMP model?

In the current study, to achieve the objectives, first a class variable is formed using the Change metric, which is the total amount of editing of source code during the maintenance phase of the software development. Further, outliers are detected and replaced by the mean values of each object-oriented predictor variable followed by standardize data. Prior to training of the SMP model using MATLAB tool, the data balancing is performed using Synthetic Minority Oversampling Technique (SMOTE) technology and WEKA tool followed by tuning of hyper-parameters. Finally, performance of each SMP is evaluated using Accuracy, Precision, Recall, and F1-Score measures calculated using confusion matrix.

The rest of the paper is organized as follows: Section 2 describes the "related work," Section 3 covers the "research methodology," Section 4 covers the "results and discussion," and finally the "conclusion" part is covered in Section 5.

II. Related Work

To accomplish our research, several works pertaining to the prediction of software maintainability are explored. It is discovered that Change metric is used as the maintainability of object-oriented software for the training of SMP models. An analysis of the techniques and maintainability metrics is summarized in Table I.

TABLE I. Maintainability and Techniques

Year	Technique	Maintainability	Reference
2014	SMPLearner	Code change history Average. maintenance Effort.	Zhang et al. [11]
2016	Support Vector Machine	Web services description language	Kumar et al. [12]
2017	Search based techniques and Hybridized techniques	Change proneness	Malhotra and Khanna [13]
2019	M5P regression tree, multilayer perceptron, multi linear regression, and support vector regression	Maintainability Index (MI)	Reddy and Ojha [14]
2020	14 Machine Learning Techniques	Change Metric	Malhotra and Lata [15]
2020	Least Square SVM	Change Metric	Gupta and Chug [16]
2020	Random Forest	Change Metric	Gupta and Chug [17]

According to Agrawal et al. 2021 [18] finding great settings, by which fine-tuning parameters that can significantly boost the accuracy of software analytics prediction, is always advantageous in terms of hyper-parameter optimization. They utilized the "DODGE" approach for hyper-parameter optimization of 120 Software Engineering data sets for defect prediction, often avoiding configurations that result in the same findings. They found that the basic DODGE works best for data sets with low intrinsic dimensionality.

Shen et al. [19] in 2020 found that the majority of developers are reluctant to invest time in looking for better design solutions because of the demand of their programming employment leading to design flaws known as "code smells" that will increase maintenance costs in the future. Developers need to be quick to spot code smells and re-factor as necessary in order to improve software quality and lower maintenance costs. They also found, based on empirical findings, that hyper-parameter adjustment can significantly increase code smell detection effectively.

In 2017, Sara et al. [20] proposed the Grid search method for tuning hyper-parameters, SMOTE technique for balancing datasets and conducted an experimental study involving five ML techniques: K-Nearest Neighbor (KNN), SVM, Decision Trees (DT), Multilayer Perceptron (MLP), and Nave Bayes (NB). The experimental results suggest that balanced data and hyper-parameter tuning are crucial for the best performance of ML algorithms.

From the literature survey, it is clear that a very few researches are published on application of SMOTE data balancing and Hyper-parameter optimization in the area of software maintainability prediction.

III. Research Methodology

Here, we proposed an algorithm for SMP which includes the formation of response variable prior to the training of SMP model. The proposed SMP algorithm is as follows:

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 2
DOI: https://doi.org/10.17762/ijritcc.v11i2.6134
***Article Received**: 26 December 2022 **Revised**: 16 January 2023 **Accepted**: 28 January 2023*

Algorithm:

Input: RD[i][j] = raw dataset; Change = vector dependent metric; i = index for instance; j = index for predictor variable; T = function of SMP; T' = function of HPOSMP; x = total machine learning methods applied;
Begin
Step 1: Formation of Dependent variable "Class":
for each instance of Dataset set class value:
if change value of the instance is greater than mean value of Change vector, then set class as Low
else High
end loop
Step 2: Formation of Dataset:
Removal of Change vector and addition of the Class vector found in step 1.
Step 3: Training of dataset using basic ML
Modelx=Tx (ND[i][j]);
Step 4: Outlier Detection and replacement
Step 5: Standardize the datasets using Z-score value.
Step 6: Data balancing using SMOTE technology
Step 7: Training of dataset using Hyperparameter optimized HPOSMP
For each ML methods
- *Selection of hyperparameters to optimize or tune.*
- *Selection of optimizer function and number of iterations to train the model.*
- *Model'x=T'x (ND[i][j]);*

End loop
Step 8: Performance evaluation:
Calculation and Comparison of performance measures of Tx and T'x built in Step 3 and Step 7.
end

A. Dataset

User Interface System (UIMS) and Quality Evaluation System (QUES) are two object oriented commercial software that we utilized to conduct the research work. Both UIMS and QUES software are written in Classic-Ada programming language [21].

B. Predictor Variable

For the training of proposed SMP models, different metrics of OO software like Depth of Inheritance tree (DIT), Lack of cohesion of methods (LCOM), Number of Children (NOC), Response for class (RFC), Data Abstraction Coupling (DAC), Message Passing Coupling (MPC), Weighted Methods per Class (WMC), Number of Methods (NOM), and two other size related metrics namely SIZE1 and SIZE2bare used as input variables in this study. The description of these object-oriented based predictor variables is given in Table II. These selected predictor metrics explain several elements such as cohesion, coupling, inheritance, size, encapsulation, and composition of OO systems.

TABLE II. Description of Input Variables

Metric Name	Abbreviation	Metric Suit	Description
DIT	Depth in the Inheritance Tree	Chidamber and Kemerer [22]	Determines the level of a class in the hierarchy of the inheritance.
LCOM	Lack of Cohesion of Methods		Count of independent local methods of a class.
NOC	Number of Children		Total number of immediate subclasses of a class
RFC	Response for Class		Count of local methods and the methods called by local methods
WMC	Weighted method complexity		Summation of all local method's McCabe's cyclomatic complexities
DAC	Data Abstraction Coupling	Li and Henry [21]	The metric that measures the coupling complexity caused by Abstract data types
MPC	Message-Passing Coupling		Count of statements sent by a class
NOM	Number of Methods		Total number methods defined in a class
SIZE1	-		Number of Semicolons Per Class
SIZE2	-		Number of Attributes + Number of Methods

C. Response variable

We have considered the change metric for the formation of dependent or response variable. Change metric describes the total number of changes made during the maintenance phase of the software development life cycle. Change metric is the count of total number of source code added, deleted and modified [11]-[17]. Class is a binary variable whose values can be either High or Low. "High" label of class variable represents the high maintainability class which requires less effort in maintenance whereas "Low" label of class variable represents the low maintainable class which requires more effort in maintenance phase. Classes for which the values of independent variables are not found were discarded.

D. Data Preprocessing

Since the model approach depends on the performance of the prediction model per dataset and response variable, this research may involve data pre-processing to detect outliers in all datasets and then replace each outlier instance with the mean value. Further, data are standardized using z-scores and data balancing is applied as per (1).

$$\mathrm{Z-score} = \frac{\mathrm{D-mean(D)}}{\mathrm{std(D)}} \quad (1)$$

Here in (1), D represents the instance, *mean ()* determines the mean value of the data vector and *std* is the standard deviation.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 2
DOI: https://doi.org/10.17762/ijritcc.v11i2.6134
***Article Received**: 26 December 2022 **Revised**: 16 January 2023 **Accepted**: 28 January 2023*

SMOTE: Malhotra and Lata [23] affirmed that in case of oversampling, minority class sample is increased in such a manner that it matches with the majority class sample and resolves the issue of data imbalance in the software maintainability prediction dataset. In the current study, SMOTE technique is used to perform the oversampling of software maintainability dataset. In SMOTE, artificial sample of minority classes from software maintainability prediction dataset is produced. For example, if the minority class instances are 100 as compared to the 1000 majority class instances, the training of the SMP model may get biased towards majority class and predict the instance as High maintainable most of the time. To overcome this problem SMOTE uses nearest neighbor approach. It first finds the minority class instances or data point and then uses the K-nearest neighbor approach to join the data points of minority class sample. To increase the minority class SMOTE creates new artificial data points at the middle of the line joining two minority class data points. This process goes on for the set number of iterations. Here, SMOTE has been applied to both the dataset QUES and UIMS using the WEKA tool.

E. Machine Learning

For statistics and machine learning total five techniques, whose compatibility is provided in the systematic literature review by Alsoli and Roper [24], have been used. A brief description of these machine learning techniques is given below:

Decision Tree (DT): A decision tree is a tree structured classifier that has two types of nodes namely decision nodes and leaf nodes. A decision node performs the test on the basis of which the next branches are formed. Thus, decision nodes are responsible for giving the direction to go from. Leaf nodes of the decision tree are categorical classes or the final result. The Gini Index is a metric that quantifies how precisely a split exists between categorized groups. The Gini index assesses a score between 0 and 1.

Logistic Regression (LR): Logistic regression is mainly used for classification problems. It works as linear regression and performs binary classification or multi class classification of continuous data. Logistic regression uses sigmoid function and fits an S-shaped curve with the range of the curve from 0 to 1. The likelihood of an event occurrence is determined through logistic regression using a set of independent variables in the data.

Gaussian Naïve Bayes (GNB): A Gaussian naïve bayes model is based on continuous variables that are assumed to have a Gaussian (or normal) distribution. Since the variables or features are independent hence the name "naïve". For software maintainability prediction, if it has to predict High or Low maintainable classes, it assumes all OO metrics as independent i.e., RFC does not depend on NOC.

Support Vector machine (SVM): Support vector machines are classifiers that locate a hyper-plane classifying the data points in an N-dimensional space (where N is the number of characteristics). The two groups of data points may be divided using a variety of different hyper-planes. It finds a plane with the maximum distance between data points from both classes. Support vector machines maximize the margin distance, which boosts the accuracy of class value predictions.

k-nearest neighbors (KNN): The k-nearest neighbors' algorithm is a supervised learning classifier that makes predictions or classifications about how a single data point will be grouped. Although it can be used to solve classification or regression problems, it is frequently used as a classification technique as it is predicated on the notion that similar points can be found nearby.

F. Hyperparameter Tunning

Hyper-parameter optimization can produce substantial results [25]. This study employs tuning of hyper-parameters to maximize the performance of SMP system in terms of Accuracy, Precision, Recall, and F1-Score. Hyper-parameter is used in cross-validation of SMP models because basic machine learning models are created using training data whereas their performance is evaluated using test data. Model parameters are the intrinsic configuration to the software maintainability prediction model where the values of these parameters can be determined by the dataset. For example, Gaussian naïve bayes is based on the parameters mean and standard deviation, K-Nearest Neighbor uses *K* as number of neighbors and weight values, Support vector machine uses support vector and standard deviations as model parameters, and in logistic regression coefficients are the model parameters. In contrast the hyper-parameter is the configuration external to the SMP model where the values of hyper-parameters cannot be determined by the dataset used for training of the model. Hyper-parameters deal with three aspects: they are set by the practitioner or data analysts, utilized to derive the model parameter, and can be derived from heuristic approach which is self-discovery of the value. In other words, the model parameters act as hyper-parameter when set by data analyst or practitioner manually.

IV. PERFORMANCE EVALUATION

In this investigation, confusion matrix [26] is used for the performance evaluation of proposed SMP models as depicted in Figure 1.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 2
DOI: https://doi.org/10.17762/ijritcc.v11i2.6134
***Article Received**: 26 December 2022 **Revised**: 16 January 2023 **Accepted**: 28 January 2023*

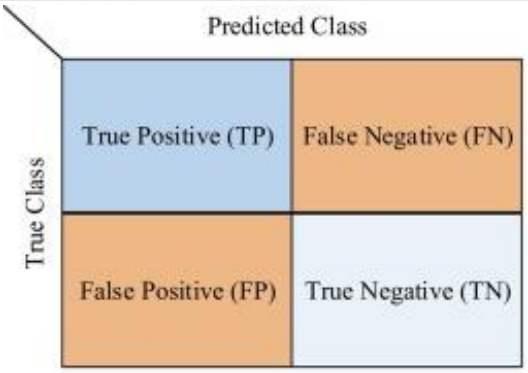

Figure 1. Confusion Matrix

Form Figure 1, it is clear that the components of a confusion matrix are True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN). For performance evaluation this study utilizes Accuracy, Precision, Recall and F1-score to compare the performance of built SMP models. The accuracy of SMP models can be calculated using (2), Precision can be calculated using (3), Recall can be calculated using (4), and F1score can be calculated using precision and recall as described in (5).

$$\text{Accuracy} = \left(\frac{TP+TN}{TP+FP+TN+FN}\right) * 100 \quad (2)$$

$$\text{Precision} = \frac{TP}{TP+FP} \quad (3)$$

$$\text{Recall} = \frac{TP}{TP+FN} \quad (4)$$

$$\text{F1} - \text{score} = 2 * \frac{\text{Precision} * \text{recall}}{(\text{Precision} + \text{Recall})} \quad (5)$$

V. RESULT AND DISSCUSSION

Based on the proposed algorithm, we performed an empirical investigation and developed and analyzed a total of 20 SMP models. Three research questions formed to achieve the objectives of our study have been responded in the following part of this section.

RQ1. What effect does data balancing have on the distribution of classes in the software maintainability datasets?

If the binary class distribution is not evenly distributed in many of the dataset, the trained SMP models can go biased towards majority class and accuracy may be degraded. We used SMOTE data balancing technique to balance the datasets. The impact of SMOTE data balancing technique is represented in Figure 2 and Figure 3 for the QUES and UIMS datasets respectively. To achieve this objective WEKA tool is utilized.

From Figure 2(a) it is evident that before data balancing the QUES dataset suffered from uneven class distribution problem. The total number of instances in QUES dataset is 71 out of which 30 and 41 classes belong to Low-class and High-class maintainability classes respectively. Figure 2(b) shows that data imbalance is removed by using SMOTE techniques whereby low majority class variables in QUES is increased to 60 resulting into a total 101 instances.

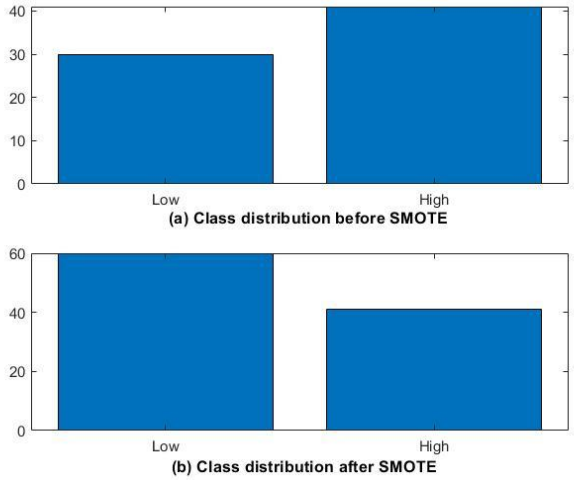

Figure 2. Class distribution before and after SMOTE on QUES

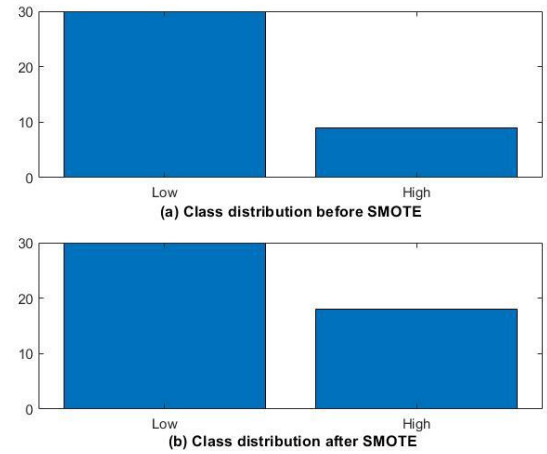

Figure 3. Class distribution before and after SMOTE on UIMS

Similarly, from Fig. 3(a) it is evident that before data balancing the UIMS dataset suffered from uneven class distribution problem. The total number of instances in UIMS dataset is 39 out of which 30 and 9 classes belong to Low and High maintainability classes respectively. Fig. 3(b) shows that data imbalance is removed by using SMOTE techniques whereby low majority class variables in QUES is increased to 18 resulting into a total 48 instances.

RQ.2: How effective are the SMP models created using HPO in comparison to the models created using basic machine learning and logistic regression?

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 2
DOI: https://doi.org/10.17762/ijritcc.v11i2.6134
***Article Received**: 26 December 2022 **Revised**: 16 January 2023 **Accepted**: 28 January 2023*

To evaluate the SMP models, we use the confusion matrix. Table 3 gives the details of the confusion matrix obtained for all the base ML techniques used for QUES and UIMS datasets. Whereas Table 4 shows the details of confusion matrix of 5 machine learning techniques after applying the hyper-parameter tuning for UIMS and QUES datasets respectively.

TABLE III. Confusion Matrix of base SMP models

Machine Learning	QUES				UIMS			
	TP	*FN*	*FP*	*TN*	*TP*	*FN*	*FP*	*TN*
Decision Tree	32	9	5	25	24	6	3	6
Logistic Regression	36	5	6	24	23	7	5	4
Gaussian Naïve Bayes	36	5	17	13	26	4	4	5
SVM	35	6	14	16	26	4	5	4
KNN	39	2	4	26	26	4	4	5

TABLE IV. Confusion Matrix of HPOSMP models

Machine Learning	QUES				UIMS			
	TP	*FN*	*FP*	*TN*	*TP*	*FN*	*FP*	*TN*
Decision Tree	35	6	6	24	27	3	5	4
Logistic Regression	36	5	6	24	23	7	5	4
Gaussian Naïve Bayes	34	7	12	18	27	3	4	5
SVM	39	2	7	23	30	0	6	3
KNN	39	2	4	26	29	1	6	3

Table 5 shows the accuracy measure of each SMP models including base ML models as well as HPOSMP models calculated using the equation (2). From the Table 5, it is evident that, in case of both the dataset i.e., QUES and UIMS, accuracy of the SMP models is substantially increased by using the proposed approach which includes a combination of SMOTE and HPO.

TABLE V. Comparison of Accuracy Measure

Machine Learning	UIMS		QUES	
	BASE SMP	*HPOSMP*	*BASE SMP*	*HPOSMP*
Decision Tree	76.9	79.5	80.3	83.1
Logistic Regression	69.2	71.5	84.5	84.5
Gaussian Naïve Bayes	79.5	82.1	69	73.2
SVM	76.9	84.6	71.8	87.3
KNN	79.5	82.1	91.5	91.5

RQ.3: What is the Precision, Recall, and F1-Score ratings for each SMP model?

The number of positive class forecasts that really fall within the positive class is measured by precision. Recall measures how many correct class predictions were produced using all of the successful cases in the dataset. Precision and recall issues are balanced in a single number by F-single Measure's score. Therefore, the performance of each SMP Model is evaluated in terms of these factors and described in Fig. 4 and Fig. 5 respectively.

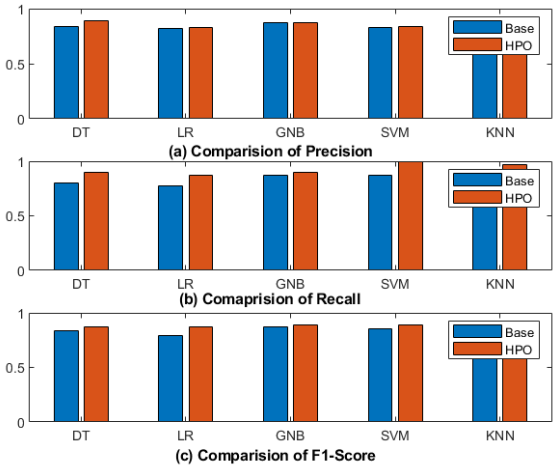

Figure 4. comparison of performance measures of UIMS dataset

Fig. 4(a), Fig. 4(b) and Fig. 4(c) represent the comparison of Precision, Recall, and F1-Score measures respectively for the UIMS dataset. It is clearly evident from the figures that the performance of each HOPSMP models is better as compared to the base SMP machine learning models. Detailed performance improvement is also shown in the Table VI.

TABLE VI. Increments in performance measures for proposed SMP using UIMS

Machine Learning	Precision	Recall	F1-Score
DT	same	12.50%	03.57%
LR	01.22%	12.99%	10.13%
GNB	same	03.45%	same
SVM	same	14.94%	04.71%
KNN	04.82%	11.49%	02.30%

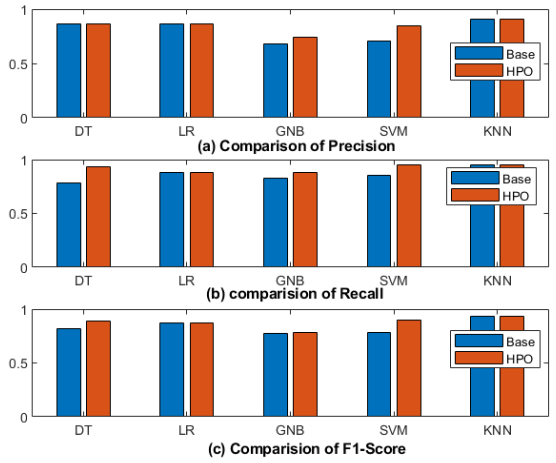

Figure 5. Comparison of performance measures of QUES dataset

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 2
DOI: https://doi.org/10.17762/ijritcc.v11i2.6134
***Article Received**: 26 December 2022 **Revised**: 16 January 2023 **Accepted**: 28 January 2023*

Similarly, Fig. 5(a), Fig. 5(b) and Fig. 5(c) represent the comparison of Precision, Recall, and F1-Score measures respectively for the QUES dataset. It is clearly evident from the figures that the performance of each HOPSMP models is better as compared to the base SMP machine learning models. Detailed performance improvement is also shown in the Table VI.

TABLE VII. Increments in performance measures for proposed SMP using QUES

Machine Learning	Precision	Recall	F1-Score
DT	same	19.23%	08.54%
LR	same	same	same
GNB	8.82%	06.02%	01.30%
SVM	19.72%	11.76%	15.38%
KNN	same	same	same

It is also concluded that Decision tree, Support Vector Machine and KNN combined with SMOTE data balancing techniques give more substantial increase in the values of performance measures in terms of Accuracy, Precision, Recall, and F1-Score as compared to Gaussian Naive bayes and Logistic Regression.

VI. Conclusion

In this research work, an algorithm (HPOSMP) for the development of effective SMP model using data balancing and hyper-parameter tuning of machine learning approach has been proposed. Datasets for software maintainability have been formed using two object-oriented software namely QUES and UIMS written in Ada-classic programming language. SMOTE technique has been used for data balancing and Decision Tree, Gaussian Naïve Bayes, K-Nearest neighbor, Logistic Regression, and Support Vector Machine are adopted as Machine Learning approaches. Initially, a response variable called Change metric is selected and then outliers are detected and dataset is standardized through SMOTE procedure for further processing. Finally, both the datasets are trained and tested using various machine learning techniques. Experimental and implementation work the research has been done using MATLAB and WEKA tools. A total of 20 SMP Models are built and analyzed.

Results were compared for performance evaluation in terms of Accuracy, Precision, Recall, and F1-Score measures for both UIMS and QUES datasets and it is found that the performance of each HOPSMP models is better as compared to the base SMP machine learning models. Further, KNN outperforms the other machine learning techniques and produces 91.5% of accuracy of SMP model. It is also concluded that Decision tree, Support Vector Machine and KNN combined with SMOTE data balancing techniques show better performance as compared to Gaussian Naive bayes and Logistic Regression techniques. Thus, the proposed HPOSMP algorithm is a great alternative for base SMP models.

This research is based on specific datasets of software written in a specific object-oriented language. Therefore, it can be further extended to explore feature selection and additional datasets supporting other programming languages like JAVA, C++ etc. Additionally, HPOSMP models can be further modified and tested using other techniques like ensembles, novel re-sampling and techniques like Boosting and Bagging.

References

[1] G. Issac, C. Rajendran, and R. N. Anantharaman, "Determinants of software quality: customer's perspective," Total Qual. Manag. Bus. Excell., vol. 14, no. 9, pp. 1053–1070, 2003.

[2] S.-H. Li, D. C. Yen, W.-H. Lu, T.-Y. Chen, "The characteristics of information system maintenance: an empirical analysis," Total Qual. Manag. Bus. Excell., vol. 25, no. 3–4, pp. 280–295, 2014.

[3] "IEEE Standard for Software Maintenance," IEEE Std 1219-1993, 1993.

[4] D. Houston, J. Bert Keats, "Cost of software quality: a means of promoting software process improvement," Qual. Eng., vol. 10, no. 3, pp. 563–573, 1998.

[5] A. Kaur, K. Kaur, "Statistical comparison of modelling methods for software maintainability prediction," Int. J. Softw. Eng. Knowl. Eng., vol. 23, no. 6, pp. 743–774, 2013.

[6] R. Malhotra, A. Chug, "Application of evolutionary algorithms for software maintainability prediction using object-oriented met- rics," in Proceedings of the 8th International Conference on Bioinspired Information and Communications Technologies, 2014, pp. 348–351.

[7] R. Malhotra, A. Chug, "Software Maintainability Prediction us- ing Machine Learning Algorithms," Softw. Eng. an Int. J., vol. 2, no. 2, pp. 19–36, 2012.

[8] R.E. Shawi, M. Maher, S. Sakr, "Automated machine learning: State-of-the-art and open challenges", arXiv preprint arXiv:1906.02287, (2019). http://arxiv. org/abs/1906.02287.

[9] M. Kuhn, K. Johnson, "Applied Predictive Modeling", Springer, 2013, ISBN: 9781461468493.

[10] G.I. Diaz, A. Fokoue-Nkoutche, G. Nannicini, H. Samulowitz, "An effective algorithm for hyperparameter optimization of neural networks", IBM J. Res. Dev. Vol. 61 pp. 1–20, 2017. https://doi.org/10.1147/JRD.2017.2709578.

[11] Zhang, W., Huang, L., Ng, V., Ge, J., "SMPLearner: learning to predict software maintainability", Automated Software Engineering, vol. 22, pp. 111-14, 2015..

[12] Kumar, L., Krishna, A., Rath, S. K., "The impact of feature selection on maintainability prediction of service-oriented applications", Service Oriented Computing and Applications, vol. 11, pp. 137-161, 2017.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 2
DOI: https://doi.org/10.17762/ijritcc.v11i2.6134
***Article Received**: 26 December 2022 **Revised**: 16 January 2023 **Accepted**: 28 January 2023*

[13] Malhotra, R., & Khanna, M., "An exploratory study for software change prediction in object-oriented systems using hybridized techniques". Automated Software Engineering, vol. 24, pp. 673-717, 2017.

[14] Reddy, B. R., Ojha, A., "Performance of Maintainability Index prediction models: a feature selection based study", Evolving Systems, vol. 10(2), pp. 179-204, 2019.

[15] Malhotra, R., Lata, K., "An empirical study to investigate the impact of data resampling techniques on the performance of class maintainability prediction models", Neurocomputing, vol. 459, pp. 432-453, 2021.

[16] Gupta, S., Chug, A., "Software maintainability prediction of open source datasets using least squares support vector machine",. Journal of Statistics and Management Systems, vol. 23(6), pp. 1011-1021, 2020.

[17] Gupta, S., Chug, A., "Software maintainability prediction using an enhanced random forest algorithm", Journal of Discrete Mathematical Sciences and Cryptography, vol. 23(2), pp. 441-449, 2020.

[18] Agrawal, X. Yang, R. Agrawal, R. Yedida, X. Shen, , and T. Menzies, "Simpler hyperparameter optimization for software analytics: why, how, when", IEEE Transactions on Software Engineering, vol. 48, pp. 2939 - 2954, Apr 2021.

[19] L. Shen, W. Liu, X. Chen, Q. Gu, and X. Liu, "Improving machine learning-based code smell detection via hyper-parameter optimization," In 2020 27th Asia-Pacific Software Engineering Conference (APSEC),IEEE, Dec 2020, pp. 276-285, doi: 10.1109/APSEC51365.2020.00036.

[20] E. Sara, C. Laila, I. Ali, "The Impact of SMOTE and Grid Search on Maintainability Prediction Models," In 2019 IEEE/ACS 16th International Conference on Computer Systems and Applications (AICCSA), IEEE, Nov 2019, pp. 1-8, doi: 10.1109/AICCSA47632.2019.9035342.

[21] Li, W., Henry, S., "Object-oriented metrics that predict maintainability", Journal of systems and software, vol. 23(2), pp. 111-122, 1993.

[22] Chidamber, S. R., Kemerer, C. F., "A metrics suite for object oriented design", IEEE Transactions on software engineering, vol. 20(6), pp. 476-493, 1994.

[23] Malhotra, R., Lata, K., "An empirical study on predictability of software maintainability using imbalanced data", Software Quality Journal, vol. 28, pp. 1581-1614, 2020.

[24] Alsolai, H., Roper, M., "A systematic literature review of machine learning techniques for software maintainability prediction.", Information and Software Technology, vol. 119, pp. 106214, 2020.

[25] Feurer, M., Hutter, F., Hyperparameter optimization. Automated machine learning: Methods, systems, challenges, 3-33, 2019

[26] Singh, P., Singh, N., Singh, K. K., & Singh, A., Diagnosing of disease using machine learning. In Machine learning and the internet of medical things in healthcare (pp. 89-111). Academic Press, 2021