







Enhancing Software Maintainability Prediction Using Multiple Linear Regression and Predictor Importance

Rohit Yadav^{1*} and Raghuraj Singh²



¹Department of Computer Science and Engineering, Dr. A.P.J. Abdul Kalam Technical University, Lucknow, Uttar Pradesh, India; ²Department of Computer Science and Engineering, Harcourt Butler Technical University, Kanpur, Uttar Pradesh, India

E-mail/Orcid Id:

RY,  rohitatknit@gmail.com,  <https://orcid.org/0000-0002-6792-8845>; RS,  rrsingh@hbtu.ac.in,  <https://orcid.org/0000-0002-1718-8324>

Article History:

Received: 28th Jun, 2023

Accepted: 11th Dec., 2023

Published: 30th Dec., 2023

Keywords:

Machine learning, Multiple linear regression, Object-oriented metric, Predictor importance, Software maintainability prediction

How to cite this Article:

Rohit Yadav and Raghuraj Singh (2023). Enhancing Software Maintainability Prediction Using Multiple Linear Regression and Predictor Importance. *International Journal of Experimental Research and Review*, 36, 135-00.

DOI:

<https://doi.org/10.52756/ijerr.2023.v36.013>

Abstract: Accurate maintenance effort and cost estimation are essential for effective software development. By identifying software modules with poor maintainability, Software Maintainability Prediction (SMP) plays a crucial role in managing software maintenance expenses. Previous research efforts have used multiple regression techniques to predict software maintainability, but the results regarding various accuracy and performance metrics are inconclusive. As such, developing a methodology that can recommend regression techniques for software maintainability prediction in the face of inconsistent performance or accuracy metrics is imperative. This research addresses the critical issue of software maintainability and presents a novel approach, the Software Maintainability Model (SMP) utilizing the Predictor Importance (PI) Method, Multiple Linear Regression (MLR), and five machine learning techniques. The proposed SMP integrates ten static source code metrics from object-oriented programming. MLR and PI implement feature selection, and the SMP's performance is evaluated based on accuracy and the Mean Magnitude of Relative Error (MMRE) parameters. Our findings are promising: for the User Interface Management System (UIMS) software, the proposed SMP demonstrates an impressive MMRE of 0.2441 and an accuracy of 91.91%. Similarly, for the Quality Evaluation System (QUES) software, an MMRE value of 0.2222 is achieved alongside a maximum accuracy of 80.95%. The ensemble method, when compared to other Machine Learning (ML) techniques, exhibits superior performance. These results affirm the effectiveness of our approach, contributing to the enhancement of software maintainability in object-oriented programming systems.

Introduction

The qualities of object-oriented (OO) software are essential for satisfying explicit and implicit requirements (Colakoglu et al., 2021). High-quality software is always a priority because of its extensive effects on many facets of life and the economy. Software maintenance is the most expensive stage of development, accounting for roughly 75% of total costs. Stressing the importance of investing in maintenance activities is essential because software bugs jeopardize functionality. According to a report from the Consortium for Information and Software Quality (Garomssa et al., 2022), only in the US, poor-coded software caused a staggering 2.08 trillion dollars in

losses in the year 2020 and 4.4 billion people were impacted by software flaws in 2016 alone, resulting in a 1.1 trillion-dollar loss to the global economy. The ISO/IEC 9126 standard has identified efficiency, functionality, maintainability, portability, reliability, and usability as characteristics of high-quality software (Jung et al., 2004). Maintainability has recently gained significant attention as a crucial quality indicator for software systems' success. Software maintainability refers to a system's or component's ability to fix flaws, enhance performance, or adapt to environmental changes. Looking more at software maintenance may affect the overall cost of software development. As a solution,



many researchers have provided various SMP models (Ahmed and Al-Jamimi, 2014; Al Dallal, 2013; Wang et al., 2019; Zhang et al., 2015). SMP forecasts the costlier classes before the maintenance phase; hence, the cost can be minimized after looking at these classes more. The SMP model needs independent and dependent metric values based on historical data of different software versions. Figure 1 illustrates the standard procedures for estimating the maintainability of any OO software. The Unified Modelling Language (UML) class diagram is used to identify the software's essential classes, and the CKJM tool (Chidember and Kemerer, 1994) calculates various OO metrics for each class. Then, using feature selection techniques, the right feature sets are chosen. Additionally, the developed SMP model uses these metric collections as inputs to predict maintainability for each distinct class of software. In the literature, various OO metrics (Mahfuz and Shill, 2023; Haner and Ercelebi, 2023; Ouellet and Badri, 2023) are used to develop the SMP model as predictor variables, whereas change metric is used as a response variable (Malhotra and Chug, 2014; Eish et al., 2015; Kumar and Rath, 2016; Kumar et al., 2017). For training of SMP models, various machine learning and statistical techniques are provided in the literature, such as Association rule mining, Bayesian networks, Clustering, Neural networks, Regression-based models, and Support Vector Machines, which are some of the widely used ML methods for maintainability prediction (Zhang et al., 2015; Kumar et al., 2017; Malhotra and Lata, 2021).

The SMP model's effectiveness depends on choosing the right metrics for object-oriented source code. The feature selection process entails selecting a suitable subset from various object-oriented programming metrics offered (Kumar and Rath, 2017; Moradi et al., 2022). Also, it still needs to be determined to develop an accurate SMP model that can predict the maintainability of a class. Keeping these facts, the following goals have been established for this study:

- To determine the significant relationship between OO Metrics and change metrics.
- Selection of an appropriate collection of OO metrics for SMP creation.
- Creation and Comparison of the proposed SMP model with existing SMPs.

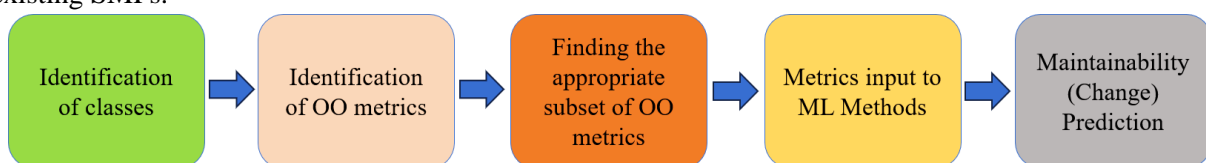


Figure 1. Flow chart for SMP model.

To achieve the objectives mentioned above, two commercial OO software are collected, and OO metrics and change metrics are extracted to build the data set. Further, MLR is applied to investigate the impact of each OO metric on the Change metric based on hypothesis testing. Additionally, PI is also used for feature selection. Three cases are created, and Five different ML techniques are applied to each case to create an SMP model. The performance of each SMP model is then evaluated using parameters such as accuracy and MMRE.

Related Work

This section gives a summary of the body of research that has been done on the use of software metrics and their applicability to SMP, as shown in Table 1. From Table 1, it can be understood that for SMP, the maintainability can be determined by the Change metric and used in various studies (Li and Henri, 1993; Koten and Gray, 2006; Zhou and Leung, 2007, 2013; Al-Jamimi, 2013) whereas maintainability as maintainability index (MI) are utilized by Zhou and Xu (2008), Coleman et al. (1994), and Zhang et al. (2015).

As Table 1 illustrates, the majority of the research that has already been done is focused on certain methods and performance indicators. The study of predictor significance and method integration are noticeably underemphasized. For instance, even though some researchers have used various methods, such as Support Vector Machines and Bayesian networks, none have specifically addressed the idea of predictor relevance. Moreover, the novel method that combines Multiple Linear Regression, Object-Oriented Metrics, and Predictor Importance is still unexplored.

Research Methodology

In order to create an effective SMP model, the current study investigates how OO metrics and the change metric relate to one another. The study focuses on three cases for feature selection and training using five machine learning techniques. The current study involves the following steps:

1. Dataset Formation: The initial dataset comprises ten OO metrics, considered predictor variables or input variables, and the change metric, the response variable representing software maintainability. These metrics

Table 1. An overview of the empirical research on maintainability

Year	Author	Maintainability	Techniques	Performance measure
1993	Li & Henry	CHANGE	Regression Analysis	R ² and Adjusted R ²
1994	Coleman et al.	Maintainability = 171 -5.2 x ln(aveVol) -0.23 x ave V(g') -16.2 x ln(aveLOC) +(50 x sin (d2.46 x perCM))	Regression Analysis	-
2006	C. van Koten, A.R. Gray	CHANGE	Bayesian network	MMRE
2007	Zhou & Leung,	CHANGE	multivariate adaptive regression splines	MMRE
2008	Zhou & Xu	Maintainability Index (MI)= 171-5.2 ln(aveV) - 0.23aveV(g') - 16.2 ln (aveLOC) + 50 sin (sqrt (2.4perCM))	Univariate and Multivariate Regression Analysis	Absolute relative error, magnitude of relative error, and R squared
2013	Al Dallal,	CHANGE	Multivariate logistic regression analysis	Precision, Recall, Inverse precision (denoted IP), Inverse Recall (denoted IR)
2013	Ahmed & Al-Jamimi,	CHANGE	Mamdani fuzzy inference engine	NRMSE, MMRE
2014	Malhotra & Chug,	CHANGE	1. Group Method of Data Handling 2. Feed Forward 3-Layer Back Propagation Network 3. General Regression Neural Network	magnitude of relative error (MARE)
2015	Elish et al.	CHANGE	Ensemble methods	MMRE
2015	Zhang et al.	MI & ME	automated tool: SMPlearner	Spearman's Rank Correlation Coefficient
2016	Kumar and Rath	CHANGE	functional link artificial neural network (FLANN) with genetic algorithm (GA), particle swarm optimization (PSO)	Standard Error of Mean, MAE, MMRE

			and clonal selection algorithm (CSA) also rough set analysis (RSA) and Principal Component Analysis (PCA)	
2017	Kumar et al.	CHANGE	Support Vector Machine	Precision, Recall, Specificity, F-Measure, AUC
2019	Wang et.al.	CHANGE	Fuzzy Network	MMRE
2020	Malhotra & Lata	change count (CC)	AB, C4.5, BAGG, IRBFNN, KNN, KS, LR, MLP-CG, RBFNN	Sensitivity, Specificity, G-mean, Balance
2021	Malhotra & Lata	CM (Class Maintainability)	28 classification techniques	g-mean (GM), and balance (BL)
2023	Kumar & Kaur	CHANGE	Multi Criteria Decision Making with 22 regression techniques	Eight Performance measures
2023	Jaya Bharath et al.	CHANGE	Gradient Boost Classifier	Accuracy
2023	Hu et al.	Static Code	Deep Neural Network (DeepM)	Accuracy

are collected from software projects to create the foundation for the research analysis.

2. Preprocessing: Before conducting the analyses, the dataset undergoes preprocessing (missing data, outlier removal, and normalization).
3. Feature Selection and Case Formation: Three distinct cases are formed for feature selection, each involving different methods for selecting relevant features to build predictive models:

Case 1: As a baseline, in this case, all ten OO metrics are used as features for both training and testing the software maintainability prediction model.

Case 2: MLR is employed as a method for hypothesis testing and feature selection.

Case 3: PI analysis is utilized for feature selection.

Training and Testing: For each of the three feature selection cases, five different machine learning techniques i.e., Ensemble (ENS), Classification Tree (CT), Naïve Bayes (NB), Discriminant Analysis (DA), and Support Vector Machine (SVM) are employed for training and testing the predictive model their brief description are as follows:

- ENS: A prediction model made up of a weighted mixture of many classification models is called a classification ensemble. Combining several classification models often improves prediction accuracy (Elish et al. et al., 2016).

- CT: also known as decision trees, are used to forecast data responses. Following the choices made in the tree through the root (starting) node up to a leaf node to anticipate a response. The answer is stored in the leaf node. 'True' or 'false' are examples of nominal replies provided by classification trees (Alsolai et al., 2020).
- NB: It refers to a group of classification techniques based on Bayes' Theorem. It's not just one algorithm; rather, it's a collection of algorithms bound together by a common premise: the notion that every pair of characteristics being classed stands alone (Kaur et al., 2014).
- DA: It makes the assumption that various groups use different Gaussian distributions to construct their data. The classifier computes the Gaussian distribution parameters for every group during training. It chooses the class with the lowest misclassification expense in order to forecast fresh data (Yenduri and Gadekallu, 2023).
- SVM: It finds the optimum hyperplane in an N-dimensional space to efficiently split data points into different classes (Gupta and Chug, 2020).

1. Performance Evaluation: comparative analysis using MMRE and accuracy.

The accuracy of the SMP model is calculated using a confusion matrix. It has four aspects: true positive (TP), true negative (TN), false positive (FP), and false negative

(FN). Consider that positive means a high-maintenance class and negative means a low-maintenance class, in such situation TP: The predicted value is positive, and the actual value is also positive. FP: The predicted value is positive, but the actual value is negative. FN: The predicted value is negative, but the actual value is positive. TN: The predicted value is negative, and the actual value is also negative. Flowchart for the proposed work is presented in Figure 2.

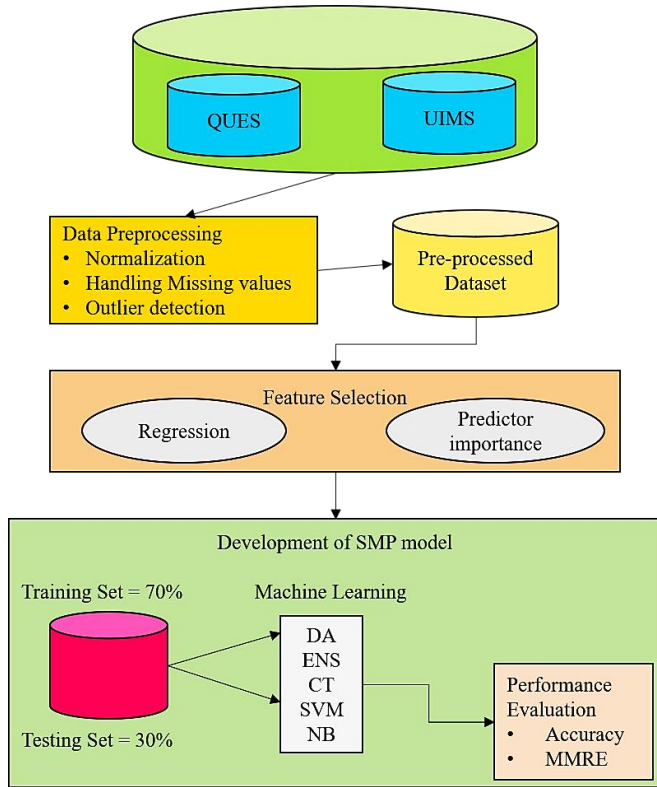


Figure 2. Flowchart of the proposed SMP model.

Experimental Setup

The SMP's creation, considering various case studies, is the primary subject of the present section. The data is normalized to increase accuracy, and both dependent and independent variables are chosen to create the SMP model.

Dataset Description

The dataset is obtained using two of the most popular commercial software systems, i.e., UIMS and QUES, which were developed using Classic-Ada programming language, hence providing the OO paradigm. UIMS contains ten OO metrics and a Change metric with 39 instances, whereas QUES contains 10 OO metrics and a Change metric with 74 instances. The dataset was proposed by Li and Henry (Li and Henry, 1993).

Dependent variable: Change metric

Authors have developed different definitions and metrics for software maintainability (Li and Henry, 1993; Zhou and Leung, 2007). According to the existing literature, most researchers use the "MI" and "change metric" to

evaluate the maintainability of software. In the current study, the change metric is considered to be maintainability. Maintainability is the amount of modification added to the code throughout maintenance. A line change is defined as either the "addition" or "deletion" of lines of code within a class during the maintenance phase (Malhotra and Chug, 2014; Li and Henry, 1993; Zhou and Leung, 2007).

Predictor/Independent Variable: OO metric

The current study focuses on ten static source code metrics within the OO paradigm. Selected OO metrics for SMP are shown in Table 2.

Table 2. List of OO Metric used in proposed SMP model

Metric suit	Metric	Object Oriented property
(Chidamber and Kemerer, 1994)	DIT	Inheritance
	LCOM	Cohesion
	NOC	Inheritance
	RFC	Coupling
	WMC	Cyclomatic complexity
(Li and Henry 1993)	DAC	Abstraction and coupling
	MPC	Coupling
	NOM	Encapsulation
	SIZE1	Elements of Source Code
	SIZE2	

DIT is the maximum length from the node to the tree's root. The cohesion and interdependence of methods in a class are assessed using LCOM in OO programming. It quantifies how much information or variables are shared between methods in a class. Indicating that the methods are less related or lack semantic coherence, a higher LCOM value suggests weaker cohesion. NOC counts the immediate subclasses of a parent class. For example, in Java programming, if classes B and C inherited a class named A, then the NOC value for class A would be 2. RFC is a metric for how many methods in a class can be used to respond to a message. It is assumed that the higher the value of RFC, the higher the complexity and the lower the effort and maintainability of the code. For example, if Class A is inherited by Class B and both have two methods, then the object of Class B can invoke all four methods: therefore, RFC (Class B) =4, whereas RFC (Class A) = 2. WMC is the average of all the complexities of methods defined in a class. If the WMC value of a class is high, it means the class is more complex and vice versa. DAC is the count of abstract data types defined in a class. DAC generally depicts that

a class has too many responsibilities. In other words, there are many fields with references. For example, if Class A declares two different objects, then the DAC (Class A) = 2. The MPC counts how often a class's methods refer to methods in other classes, indicating how dependent local methods are on methods implemented by other classes. It enables analysis of the message transmission (method calls) between the involved classes' objects. NOM counts the total number of public methods defined in a class. For example, if two methods are public in a class A, one is private, and one is protected, then NOM (Class A) =2. SIZE1 determines the number of semicolons used, and SIZE2 determines the addition of several attributes and methods in a class.

Efficiency of metrics

In the current study, three cases are defined for selecting different sets of OO metrics, as described in Table 3. Based on three cases, c1, c2, and c3, SMP is constructed, and performance is evaluated.

Table 3. Test case description.

Case	Response variable	Predictor variables
c1	Change metric	DIT, WMC, RFC, NOC, DAC, LCOM, MPC, NOM, SIZE2, SIZE1
c2	Change metric	Reduced feature attributes using MLR
c3	Change metric	Reduced feature attributes using PI

Feature selection

A crucial step in predicting maintainability is choosing the appropriate software metric set. This study uses two different kinds of feature selection methods to increase the predictability of OO software maintainability. These methods assist in selecting the relevant group OO metrics from the more extensive selection of options.

MLR

Regression analysis estimates how a dependent variable will change as an independent variable or group of independent variables changes. In the current study, MLR (Riaz et al., 2009) is utilized in order to analyze how strong the relationship is between various OO metrics and change. MLR can be demonstrated by Equation (1).

$$M = b_0 + b_1 * X_1 + b_2 * X_2 + b_3 * X_3 \dots\dots\dots (1)$$

In Equation (1), M represents the dependent variable, X1, X2, and X3 are the independent variables, and b0, b1, b2, and b3 are the coefficient values. Four different properties of MLR include Coefficient Estimates (CE),

Standard error (SE), t-statistic(t-stat), and p-value. Hypothesis testing and feature selection process may be based on the p-value of the t-statistic. For instance, if the p-value of X2 in Equation (1) is more significant than 0.05, this term is insignificant at the 5% significance level given the other terms in the model and vice-versa. The workflow of MLR is shown in Figure 3.

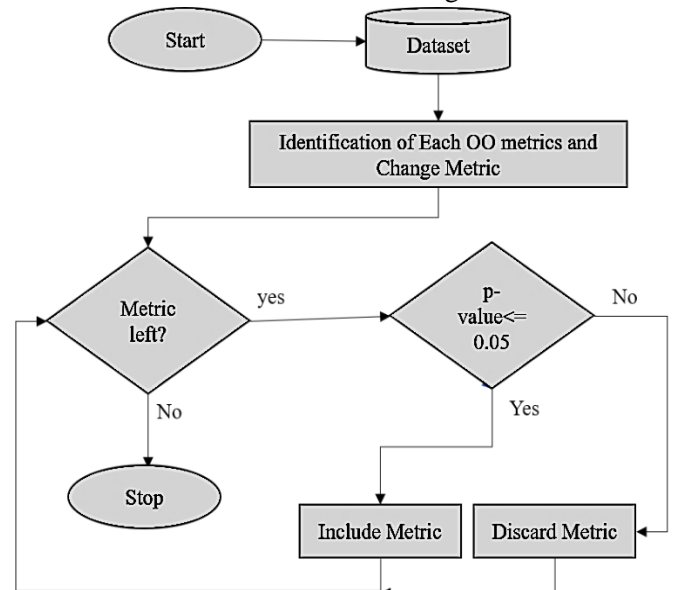


Figure 3. Feature Selection using MLR.

PI

PI is another method used in the current study to assess the significance of each predictor of a tree (Xu et al., 2019). It involves evaluating the impact of splits for each predictor on node risk and summing up these changes. The total sum is then divided by the number of branch nodes. The difference between the risk of the parent node and the combined risk of its two offspring nodes signifies the actual change in node risk caused by the predictor splits. To illustrate, when a tree divides a parent node (let us say node 1) into two child nodes (for instance, nodes 2 and 3), the significance of the split predictor is enhanced according to equation 2 in the PI method.

$$S_1 = (R_1 - R_2 - R_3) / Nbranch \dots\dots\dots (2)$$

R_i represents the node risk of node i, and Nbranch is the overall count of branch nodes. S₁ is the significance of node 1. A node's risk is determined by its error or impurity, which is then weighted by the probability associated with that particular node.

$$R_i = P_i * E_i \dots\dots\dots (3)$$

Here, P_i denotes the node probability of node i, and E_i corresponds to either the node error (in the case of a tree grown using the towing criterion) or node impurity (for a tree grown using an impurity criterion like the Gini index or deviance) for that specific node i.

Performance Evaluation

Since the SMP belongs to the prediction models, Accuracy and MMRE (Elish et al., 2015; Kumar & Rath, 2016; Wang et al., 2019) are the two parameters used for the performance evaluation. The formula of the Accuracy measure is defined in Equation (4). MMRE is used in this study to compare results with earlier investigations. MMRE stands for a mean of a measure called the magnitude of relative error. Because of the linearity of the mean, any measure that reduces the predicted magnitude of relative error (MRE) would also reduce the expected MMRE. Equation (5) can be calculated where X_i' represents the predicted outcome, X_i is the actual outcome, and n is the total number of observations.

$$\text{Accuracy (ACC)} = \frac{TP+TN}{TP+FP+TN+FN} \dots\dots\dots (4)$$

$$\text{MMRE} = \frac{1}{n} \sum_{i=1}^n \frac{|X_i' - X_i|}{X_i} \dots\dots\dots (5)$$

Results and Discussion

This section describes the results obtained by multiple regression analysis and hypothesis testing, followed by the performance analysis of SMP for each case. Multiple linear regression predicts a variable using multiple predictors. When used for binary classification, it can be adapted by setting a threshold on predicted values for categorization. Hypothesis testing assesses each predictor's significance and their combined impact on the outcome, often determined using methods like p-values or F-tests.

Results of Multiple Linear Regression

This section presents the results of a regression analysis conducted on the UIMS and QUES software dataset to investigate the impact of various OO metrics on the Change metric as described in Table 4. The analysis aimed to discern the impact of different metrics on SMP performance and provide insights into the significance and direction of feature selection. Based on the p-value, the following null and alternate hypotheses are considered in the present work.

H_0 : "No significant correlation exists between OO metric and Change Metric."

H_a : "There is a significant correlation between OO metric and Change Metric."

When considering UIMS, Table 4 shows that the intercept term in the regression model was estimated to be 9.5395 with a standard error of 30.802. The associated t-value of 0.3097 resulted in a p-value of 0.75908. However, the intercept's p-value suggests that it is not statistically significant, indicating that it may not sub-

stantially influence the change metric. The regression analysis and NOC metric have a coefficient estimate of 10.705, accompanied by a standard error of 4.2832. The computed t-value of 2.4993 led to a p-value of 0.018582. These results suggest that the NOC metric has a statistically significant positive effect on UIMS performance, hence rejecting the null hypothesis. The LCOM metric had a coefficient estimate of 4.4209, and its standard error was 2.3058. The t-value of 1.9173 resulted in a p-value of 0.065457. This implies that the LCOM metric has a marginally significant favorable influence on the change metric, hence rejecting the null hypothesis. In the WMC metric, the coefficient estimate was 4.2741, and the standard error was 2.7144. The t-value of 1.5746 corresponded to a p-value of 0.12658. This suggests that the WMC metric has a marginally significant positive effect on UIMS performance, hence rejecting the null hypothesis. Since the p-values for DIT, MPC, RFC, DAC, NOM, SIZE1, and SIZE2 metrics are less than 0.05, these metrics do not significantly impact change metrics. Therefore, NOC, LCOM, and WMC are selected as the subset of SMP features for UIMS.

When considering QUES, Table 4 shows that the intercept term in the regression model was estimated to be -3.4033 with a standard error of 16.365. The corresponding t-value of -0.20796 resulted in a p-value of 0.83597. These findings indicate that the intercept term is not statistically significant, suggesting that it may not significantly contribute to the variation in the QUES scores. For LCOM, DAC, and WMC, the p-values are 0.0085661, 0.00094389, and 0.0032928, respectively. This implies that these metrics have a statistically significant adverse effect on the Change metric and also reject the null hypothesis for these metrics. P-values for SIZE1 and SIZE2 metrics are 1.2129e-08 and 0.095063, respectively. This reveals that SIZE1 has a highly significant positive effect on the Change metric, and SIZE2 has a marginally significant positive effect on the Change metric, resulting in rejecting the null hypothesis. Also, the result of regression analysis on QUES suggests that p-values for DIT, MPC, RFC, and NOM metrics are more significant than 0.05; therefore, these metrics show no impact on the change metric. As far as NOC is concerned, since the value of 0 results in a coefficient estimate of 0 and a standard error of 0. Since NOC is constant, its t-value and p-value are Not-a-Number. This implies that the NOC predictor cannot contribute meaningfully to the regression analysis due to its constant nature.

Table 4. MLR Statistics for UIMS and QUES.

Metric	UIMS				QUES			
	Estimate	SE	t-value	P-value	Estimate	SE	t-value	P-value
(Intercept)	9.5395	30.802	0.3097	0.75908	-3.4033	16.365	-	0.83597
DIT	-3.5758	11.095	-0.32229	0.74963	5.9466	7.4352	0.79979	0.42699
NOC	10.705	4.2832	2.4993	0.018582	0	0	NaN	NaN
MPC	2.7355	6.0698	0.45067	0.6557	-0.68361	0.67612	-1.0111	0.31603
RFC	-0.92205	2.5565	-0.36067	0.72105	0.45772	0.32204	1.4213	0.16041
LCOM	4.4209	2.3058	1.9173	0.065457	-3.8794	1.4271	-2.7183	0.0085661
DAC	11.21	15.377	0.729	0.47205	-13.841	3.9786	-3.4789	0.00094389
WMC	4.2741	2.7144	1.5746	0.12658	-1.4923	0.48743	-3.0615	0.0032928
NOM	-2.5711	16.751	-0.15349	0.87911	-4.6262	3.6341	-1.273	0.20793
SIZE2	-0.25786	15.971	-0.01615	0.98723	6.1088	3.6018	1.696	0.095063
SIZE1	-0.29626	0.38651	-0.76651	0.44979	0.36601	0.055481	6.597	1.2129e-08

Results of Predictor Importance

Figure 4 shows the importance of each OO metric for predicting the Change metric. The horizontal axis represents each OO metric, and the vertical axis represents the estimated values. It is observed that RFC, NOC, MPC, DAC, NOM, SIZE1, WMC, DIT, SIZE2, and LCOM are top-ranked predictor variables in this order that have a high impact on the Change of OO software system UIMS, whereas in the case of QUES, the decreasing order of importance of predictors is SIZE2, MPC, WMC, SIZE1, DAC, NOM, LCOM, RFC, and DIT. The selected features for all the cases C1, C2, and C3 presented in the current study are shown in Table 5.

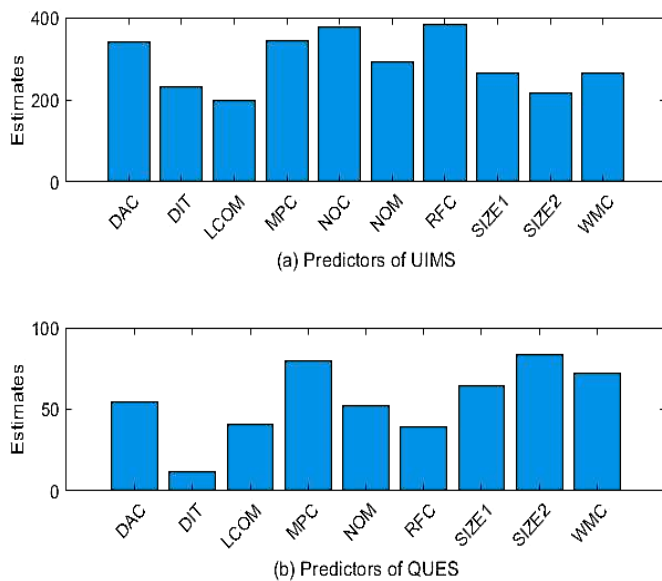


Figure 4 (a & B). PI estimates for SMP.

Table 5. Selected Features for SMP in all cases.

Case	UIMS	QUES
C1	DIT, NOC, RFC, LCOM, WMC, MPC, DAC, NOM, SIZE1, SIZE2	DIT, NOC, RFC, LCOM, WMC, MPC, DAC, NOM, SIZE1, SIZE2
C2	NOC, LCOM, WMC	LOC, DAC, WMC, SIZE1, SIZE2
C3	DAC, MPC, NOC, RFC, SIZE1, SIZE2 and WMC	DAC, MPC, NOM, WMC, SIZE1, and SIZE2

Comparative Analysis

This section presents the results of a comprehensive evaluation of SMP methods for UIMS and QUES in various cases (C1, C2, and C3). The evaluation is based on two performance metrics, i.e., MMRE and Accuracy. The analysis aims to identify the most effective SMP method for each case. Table 6 shows the results of each case C1, C2, and C3 for UIMS. It is evident from Table 6 That Case C1 shows that CT, NB, and SVM achieved comparable MMRE values of 0.2500. CT and NB also exhibited high accuracy rates of 81.82%. Case C2, CT, and SVM yielded MMRE values of 0.2441 and high accuracy rates of 91.91%. This indicates that CT and SVM outperformed other methods, delivering accurate and consistent predictions for UIMS analysis in Case C2. For Case C3, NB and SVM achieved MMRE values of 0.2441, along with accuracy rates of 91.91%. These results indicate that NB and SVM are well-suited for UIMS analysis in Case C3, demonstrating accurate predictions and minimal magnitude of relative error.

Table 6. MMRE and ACCURACY of each utilized method for UIMS for each case.

Method	C1		C2		C3	
	MMRE	ACCURACY	MMRE	ACCURACY	MMRE	ACCURACY
CT	0.2500	81.82	0.2441	91.91	0.2441	91.91
ENS	0.4167	54.55	0.3550	81.82	0.3550	81.82
DA	0.3056	81.82	0.4167	90.91	0.4167	90.91
NB	0.2500	81.82	0.2500	81.82	0.2500	81.82
SVM	0.3056	63.64	0.2441	91.91	0.2441	91.91

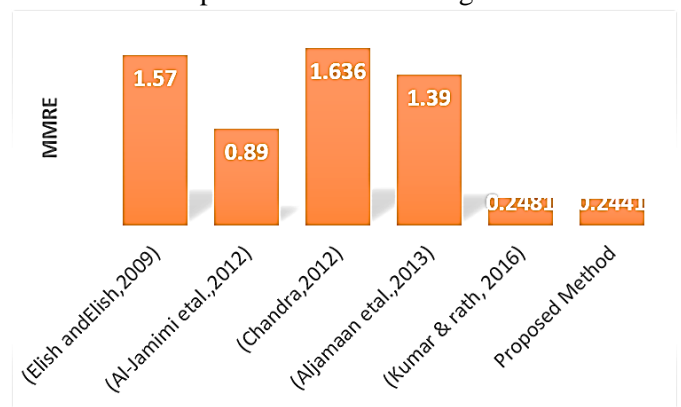
The evaluation of different classification methods for assessing the QUES across distinct cases (C1, C2, and C3) is summarized in Table 7. Across all cases, CT and ENS methods consistently achieved similar MMRE values of 0.2778 and demonstrated an accuracy rate of 76.19%. Regardless of the specific case, these methods showcase stable and reliable performance in QUES analysis. DA exhibited varying MMRE and accuracy scores across the cases. Notably, in Case C1, DA yielded an MMRE of 0.3550 and an accuracy rate of 66.67%. However, DA's performance deteriorated in Cases C2 and C3, resulting in higher MMRE values and lower accuracy rates of 0.4861 and 57.14%, respectively. This suggests that the effectiveness of DA is contingent on the particular QUES context. NB and SVM consistently demonstrated moderate to low-performance accuracy, yielding the SVM's highest MMRE values of 0.5139 and accuracy rates of 52.38%. On the other hand, SVM's performance remained relatively stable, with MMRE values ranging from 0.3056 to 0.3194 and accuracy rates of 71.43%.

Table 7. MMRE and ACCURACY of each utilized method for QUES for each case.

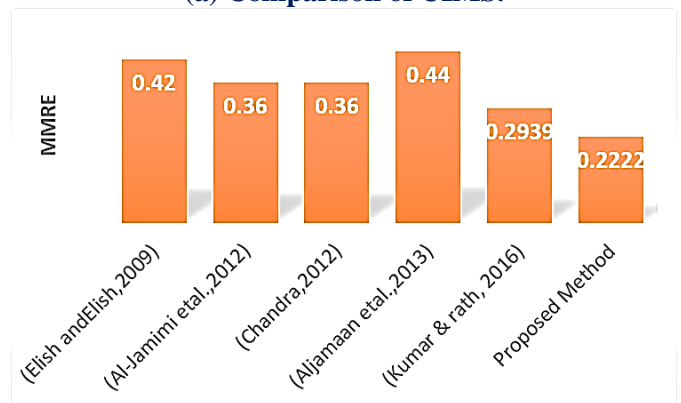
Method	C1		C2		C3	
	MMRE	ACCURACY	MMRE	ACCURACY	MMRE	ACCURACY
CT	0.2778	76.19	0.2778	76.19	0.3333	71.43
ENS	0.2778	76.19	0.2222	80.95	0.2222	76.19
DA	0.3550	66.67	0.4861	57.14	0.4861	57.38
NB	0.5139	52.38	0.5139	52.38	0.5139	52.38
SVM	0.3194	71.43	0.3194	71.43	0.3056	71.43

The best SMP models for analyzing UIMS and QUES are chosen depending on the specific scenario and preferred performance criteria. It is evident that ENS and DA consistently produce dependable forecasts for both UIMS and QUES results. According to the overall trend of the findings, these figures unequivocally show that ENS consistently outperforms alternative SMP models in terms of predictive capability.

Additionally, for comparative analysis with existing studies (Elish and Elish, 2009; Al-Jamimi, 2012; Chandra, 2012; Aljamaan, 2013; Kumar and Rath, 2016), MMRE values are compared. The studies selected used the same dataset and performance measure, i.e., MMRE. It was found that the combination of CT and PI gives a 0.2441 value of MMRE for UIMS, which outperforms the other, whereas the combination of ENS and PI outperforms the others. The Comparison is shown in Figure 5.



(a) Comparison of UIMS.



(b) Comparison of QUES.

Figure 5 (a & b). Error Comparison of the proposed method with previous studies.

The research findings shown in Tables 6 and 7 highlight the potential for the suggested approaches to play a significant role in the maintainability of software systems in practice. A comparison to previous studies in Figure 5 further supports this. Improved accuracy and decreased prediction mistakes are key features of these approaches for practical use, and decision-making for finding the low-maintenance classes and working on

them will result in improving the overall cost of software development.

Overall, the computational complexity could be influenced by the dataset size, the number of features, and the algorithm's intricacy in each methodology step. The scalability of these computations could become a concern when dealing with large-scale datasets or when applying resource-intensive algorithms.

Conclusion

The popular UIMS and QUES software datasets were thoroughly examined in this study, largely focused on improving SMP accuracy and assessing the effects of several OO metrics on the Change metric. The current research emphasises building an SMP model integrating MLR, PI and five ML techniques. MLR and PI are used for feature selection in the proposed SMP model. Based on MLR analysis on UIMS, NOC displayed a statistically meaningful positive effect on UIMS performance among the metrics, supported by its coefficient estimate of 10.705 and a notably low p-value of 0.018582. Similarly, LCOM exhibited a subtly significant positive influence, whereas WMC demonstrated a marginal yet noteworthy positive impact. For QUES, the outcomes underscored that the intercept term lacked statistical significance, implying its limited contribution to the variability in QUES scores. LCOM, DAC, and WMC significantly showcased statistically significant adverse effects on the Change metric. In addition, SIZE1 emerged with a highly significant positive influence, while SIZE2 registered a slight but discernible positive impact. In contrast, PI analysis underscored the substantial influence of metrics such as DAC, MPC, NOC, RFC, SIZE1, SIZE2, and WMC for UIMS and DAC, MPC, NOM, WMC, SIZE1, and SIZE2 for QUES. The comprehensive evaluation of SMP methods for both UIMS and QUES consistently revealed the strong performance of CT and ENS methods. These outcomes showcased their effectiveness across various scenarios, yielding precise predictions and low MMRE values. The study's findings demonstrate how defining important criteria for QUES and UIMS may offer practical guidance to practitioners in selecting the best SMP approaches for improved user experience and software quality. Additionally, showcasing the synergy between MLR, PI, and several ML techniques in the SMP model suggests that these methods might be included in additional software quality evaluation studies. The study's limitations include its focus on specific datasets, which may limit generalizability and the fact that statistical significance does not guarantee practical impact. Future research might examine longitudinal changes in metrics

and SMP techniques, as well as corroborate findings using a range of software datasets to enhance the SMP model's accuracy and usefulness in assessing software quality. In summary, this research highlighted influential metrics for UIMS and QUES, offering valuable guidance to practitioners when selecting SMP methods and conducting feature selection to enhance software quality and user experience. Further validation and practical implementation of these findings in real-world software contexts are highly recommended.

Acknowledgement

The authors wish to acknowledge Dr. A.P.J. Abdul Kalam Technical University, Lucknow, Uttar Pradesh (226021), INDIA, for utilizing the various facilities to carry out this research.

Conflict of interest

There is no conflict of interest to disclose.

References

- Ahmed, M. A., & Al-Jamimi, H. A. (2013). Machine learning approaches for predicting software maintainability: a fuzzy-based transparent model. *IET Software*, 7(6), 317-326. <https://doi.org/10.1049/iet-sen.2013.0046>
- Al Dallal, J. (2013). OO class maintainability prediction using internal quality attributes. *Information and Software Technology*, 55(11), 2028-2048. <https://doi.org/10.1016/j.infsof.2013.07.005>
- Aljamaan, H., Elish, M. O., & Ahmad, I. (2013). An ensemble of computational intelligence models for software maintenance effort prediction. Springer Berlin Heidelberg. In *Advances in Computational Intelligence: 12th International Work-Conference on Artificial Neural Networks, IWANN 2013, Puerto de la Cruz, Tenerife, Spain, June 12-14, 2013, Proceedings, Part I 12*, pp. 592-603. https://doi.org/10.1007/978-3-642-38679-4_60
- Al-Jamimi, H. A., & Ahmed, M. (2012, June). Prediction of software maintainability using fuzzy logic. In *2012 IEEE International Conference on Computer Science and Automation Engineering*, pp. 702-705. <https://doi.org/10.1109/ICSESS.2012.6269563>
- Alsolai, H., & Roper, M. (2020). A systematic literature review of machine learning techniques for software maintainability prediction. *Information and Software Technology*, 119, 106214. <https://doi.org/10.1016/j.infsof.2019.106214>
- Chandra, D. (2012). Support vector approach using a radial kernel function to predict software

- maintenance effort based on a multivariate approach. *International Journal of Computer Applications*, 51(4), 21–25.
<https://doi.org/10.5120/8029-1302>
- Chidamber, S. R., & Kemerer, C. F. (1994). A metrics suite for OO design. *IEEE Transactions on Software Engineering*, 20(6), 476-493.
<https://doi.org/10.1109/32.295895>
- Colakoglu, F. N., Yazici, A., & Mishra, A. (2021). Software product quality metrics: A systematic mapping study. *IEEE Access*, 9, 44647-44670.
<https://doi.org/10.1109/ACCESS.2021.3054730>
- Coleman, D., Ash, D., Lowther, B., & Oman, P. (1994). Using metrics to evaluate software system maintainability. *Computer*, 27(8), 44–49.
<https://doi.org/10.1109/2.303623>
- Elish, M. O., Aljamaan, H., & Ahmad, I. (2015). Three empirical studies on predicting software maintainability using ensemble methods. *Soft Computing*, 19, 2511-2524.
<https://doi.org/10.1007/s00500-014-1576-2>
- Garomssa, S. D., Kannan, R., Chai, I., & Riehle, D. (2022). How software quality mediates the impact of intellectual capital on commercial open-source software company success. *IEEE Access*, 10, 46490-46503.
<https://doi.org/10.1109/ACCESS.2022.3170058>
- Gupta, S., & Chug, A. (2020). Software maintainability prediction of open source datasets using least squares support vector machines. *Journal of Statistics and Management Systems*, 23(6), 1011–1021.
<https://doi.org/10.1080/09720510.2020.1799501>
- Haner Kirg il, E. N., & Er eplebi Ayyıldız, T. (2023). Predicting Software Cohesion Metrics with Machine Learning Techniques. *Applied Sciences*, 13(6), 3722.
<https://doi.org/10.3390/app13063722>
- Hu, Y., Jiang, H., & Hu, Z. (2023). Measuring code maintainability with deep neural networks. *Frontiers of Computer Science*, 17(6), 176214.
- JayaBharath, M., Choudary, N. L., Pranay, C. S., Praveenya, M. D., & Reddy, B. R. (2023, March). An analysis of Software Maintainability Prediction Using Ensemble Learning Algorithms. IEEE. In 2023 3rd International Conference on Artificial Intelligence and Signal Processing (AISP), pp. 1-5.
- Jung, H. W., Kim, S. G., & Chung, C. S. (2004). Measuring software product quality: A survey of ISO/IEC 9126. *IEEE Software*, 21(5), 88-92.
<https://doi.org/10.1109/MS.2004.1331309>
- Kaur, A., Kaur, K., & Pathak, K. (2014, September). Software maintainability prediction by data mining of software code metrics. IEEE, In 2014, there was an International Conference on Data Mining and Intelligent Computing (ICDMIC), pp. 1-6.
<https://doi.org/10.1109/ICDMIC.2014.6954262>
- Kumar, A., & Kaur, K. (2023). Recommendation of Regression Techniques for Software Maintainability Prediction with Multi-Criteria Decision-Making. *International Journal of Information Technology & Decision Making*, 22(03), 1061-1105.
- Kumar, L., & Rath, S. K. (2016). Hybrid functional link artificial neural network approach for predicting maintainability of OO software. *Journal of Systems and Software*, 121, 170–190.
<https://doi.org/10.1016/j.jss.2016.01.003>
- Kumar, L., & Rath, S.K. (2017). Software maintainability prediction uses a hybrid neural network and fuzzy logic approach with a parallel computing concept. *International Journal of System Assurance Engineering and Management*, 8, 1487–1502.
<https://doi.org/10.1007/s13198-017-0618-4>
- Kumar, L., Krishna, A., & Rath, S. K. (2017). The impact of feature selection on maintainability prediction of service-oriented applications. *Service Oriented Computing and Applications*, 11, 137–161.
<https://doi.org/10.1007/s11761-016-0202-9>
- Kumar, L., Kumar, M., & Rath, S. K. (2017). Maintainability prediction of web service using support vector machine with various kernel methods. *International Journal of System Assurance Engineering and Management*, 8, 205-222. <https://doi.org/10.1007/s13198-016-0415-5>
- Li, W., & Henry, S. (1993). OO metrics that predict maintainability. *Journal of Systems and Software*, 23(2), 111-122.
[https://doi.org/10.1016/0164-1212\(93\)90077-B](https://doi.org/10.1016/0164-1212(93)90077-B)
- Mahfuz, N., & Shill, P. C. (2023, February). Faulty Classes Prediction in Object-Oriented Programming Using Composed Dagging Technique. IEEE. In 2023 International Conference on Electrical, Computer and Communication Engineering ECCE), pp. 1-5.
<https://doi.org/10.1109/ECCE57851.2023.10101655>
- Malhotra, R., & Chug, A. (2014). Application of group method of data handling model for software maintainability prediction using object-oriented systems. *International Journal of System*

- Assurance Engineering and Management*, 5, 165-173.
<https://doi.org/10.1007/s13198-014-0227-4>
- Malhotra, R., & Lata, K. (2020). An empirical study on predictability of software maintainability using imbalanced data. *Software Quality Journal*, 28, 1581-1614. <https://doi.org/10.1007/s11219-020-09525-y>
- Malhotra, R., & Lata, K. (2021). An empirical study to investigate the impact of data resampling techniques on the performance of class maintainability prediction models. *Neurocomputing*, 459, 432-453.
<https://doi.org/10.1016/j.neucom.2020.01.120>
- Moradi, M., Ahmadi, M., & Nikbazm, R. (2022). Comparison of machine learning techniques for VNF resource requirements prediction in NFV. *Journal of Network and Systems Management*, 30, 1-29.
<https://doi.org/10.1007/s10922-021-09629-1>
- Ouellet, A., & Badri, M. (2023). Combining object-oriented metrics and centrality measures to predict faults in object-oriented software: An empirical validation. *Journal of Software: Evolution and Process*, e2548.
<https://doi.org/10.1002/smr.2548>
- Van Koten, C., & Gray, A. R. (2006). An application of the Bayesian network for predicting OO software maintainability. *Information and Software Technology*, 48(1), 59-67.
<https://doi.org/10.1016/j.infsof.2005.03.002>
- Wang, X., Gegov, A., Farzad, A., Chen, Y., & Hu, Q. (2019). Fuzzy network-based framework for software maintainability prediction. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 27(05), 841-862.
<https://doi.org/10.1142/S0218488519500375>
- Xu, Y., Lu, L., E, L. N., Lian, W., Yang, H., Schwartz, L. H., ... & Zhao, B. (2019). Application of radiomics in predicting the malignancy of pulmonary nodules in different sizes. *American Journal of Roentgenology*, 213(6), 1213-1220.
<https://doi.org/10.2214/AJR.19.21490>
- Yenduri, G., & Gadekallu, T. R. (2023). Xai for maintainability prediction of software-defined networks. In *Proceedings of the 24th International Conference on Distributed Computing and Networking*, pp. 402-406.
<https://doi.org/10.1145/3571306.3571443>
- Zhang, W., Huang, L., Ng, V., & Ge, J. (2015). SMP Learner: learning to predict software maintainability. *Automated Software Engineering*, 22, 111-141.
<https://doi.org/10.1007/s10515-014-0161-3>
- Zhou, Y., & Leung, H. (2007). Predicting OO software maintainability using multivariate adaptive regression splines. *Journal of Systems and Software*, 80(8), 1349-1361.
<https://doi.org/10.1016/j.jss.2006.10.049>
- Zhou, Y., & Xu, B. (2008). Predicting the maintainability of open source software using design metrics. *Wuhan University Journal of Natural Sciences*, 13(1), 14-20. <https://doi.org/10.1007/s11859-008-0104-6>

How to cite this Article:

Rohit Yadav and Raghuraj Singh (2023). Enhancing Software Maintainability Prediction Using Multiple Linear Regression and Predictor Importance. *International Journal of Experimental Research and Review*, 36, 135-146.

DOI : <https://doi.org/10.52756/ijerr.2023.v36.013>



This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.