Neurocomputing 459 (2021) 432–453

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

An empirical study to investigate the impact of data resampling techniques on the performance of class maintainability prediction models

Ruchika Malhotra [a,*], Kusum Lata [b,c]

[a] Discipline of Software Engineering Department of Computer Science & Engineering, Delhi Technological University, Delhi, India
[b] Department of Computer Science & Engineering, Delhi Technological University, Delhi, India
[c] University School of Management & Entrepreneurship, Delhi Technological University, Delhi, India

ARTICLE INFO

Article history:
Received 15 June 2019
Revised 4 January 2020
Accepted 30 January 2020
Available online 1 August 2020
Communicated by Dr. Swagatam Das

Keywords:
Maintainability prediction
Imbalanced data
Data resampling techniques
Machine learning techniques
Search-based techniques
Object-oriented metrics

ABSTRACT

With the increasing complexity of the software systems nowadays, the trend has been shifted to object-oriented (OO) development. The classes are the central construct in an OO software that are expected to be of utmost quality and high maintainability. The maintainability of a class is the probability that a class can be effortlessly modifiable in the maintenance phase. Unfortunately, it is very tough to determine the maintainability of a class with confidence before the release of the software. However, maintainability can be predicted with the help of internal quality attributes (viz. complexity, cohesion coupling, inheritance, etc.). The researchers in the literature have studied the relation amongst the internal quality attributes and class maintainability. Many class maintainability prediction models have been developed in the past with the help of internal quality attributes. Effective prediction models are vital to forecast class maintainability accurately. However, various datasets used to build prediction models for class maintainability suffer from imbalanced data problem. In that scenario, a model trained with imbalanced data gives erroneous predictions of class maintainability, which results in the inaccurate allocation of testing and maintenance resources to the misclassified classes. Therefore towards this direction, this study assesses the applicability of techniques to take care of imbalanced data. In this study the imbalanced data is treated with nine oversampling and three undersampling methods. A comprehensive comparison of fourteen machine learning (ML) techniques and fourteen search based (SB) techniques is conducted for class maintainability prediction. The results of the study support the applicability Safe-Level Synthetic Minority Oversampling Technique (Safe-SMOTE) to handle the imbalanced data for class maintainability prediction.

1. Introduction

With the ever-increasing demands of new software applications based on cutting edge technologies, the complexity of software systems is increasing day by day. The development of maintainable software is becoming a challenge for the software programming community. Maintenance is a significant phase in the software development life cycle that initiates after the release of the software product. The cost of software maintenance is approximated to be 80% of the total cost of software development, and with the increasing complexity of software systems, this cost can be even more [1]. It is very uneconomical and impractical to develop software that cannot be modified easily. So, the development of maintainable software systems substantially reduces the future maintenance cost and time. To lower down the cost of software maintenance, the earlier phases have to be designed in such a way that the final source code has better understandability and modifiability [2].

Nowadays, software development has shifted towards OO software methodology. The belief behind this shift is that OO source code has high maintainability, better understandability, and quality [3]. The classes are the fundamental units in OO software development that are expected to have high quality and enhanced maintainability. The maintainability of a class is the ease with it can be modified [4]. The class maintainability is a significant quality attribute. The classes that undergo several revisions in their lines of source code (SLOC) are said to be having less maintainability (i.e., more challenging to modify in the maintenance phase) as

* Corresponding author.
E-mail addresses: ruchikamalhotra@dtu.ac.in (R. Malhotra), kusumlata@dtu.ac.in (K. Lata).

https://doi.org/10.1016/j.neucom.2020.01.120
0925-2312/© 2020 Elsevier B.V. All rights reserved.

compared to the classes that undergo few revisions [5]. Therefore, class maintainability is a fundamental dimension of the quality of a class like other dimensions (e.g., reusability, reliability, testability, etc.) [6]. It is referred to as external software quality attribute that is the primary concern of software developers and practitioners [7]. The classes with low maintainability must be precisely tested in the testing phase to get rid of future faults [8]. Also, such classes should be well documented and refactored before the release of the software. Good documentation helps to understand the software during the maintenance better. The refactoring of the low maintainability classes by their developers in the initial stages is more effective than the refactoring done by the third party maintenance persons because the developers who develop the product are more knowledgeable and sensible about the systems. So, refactoring of low maintainability classes done by the developers saves the cost of maintenance of software. Therefore, the software practitioners are fascinated towards figuring out the low maintainability classes before the actual releases of the software so that the quality of such classes can be enhanced before the software product reaches in the hands of the customers. Unfortunately, it is challenging to measure the class maintainability before the actual maintenance. However, by then, it is very much difficult to have control over the cost of software maintenance. But class maintainability can be predicted as various internal quality attributes (e.g., inheritance, cohesion, coupling, size, etc.) can be determined after the development of the software. These attributes ought to have a relationship with the external software quality attributes, including class maintainability [9]. There are many studies in literature advocating the prediction of maintainability by establishing a relation between the internal quality attributes with the class maintainability [10–13]. Various prediction models have been developed in the past literature to forecast the class maintainability in the earlier phases of software development by using the historical datasets and different ML, statistical, and hybridized techniques [5,12,14,15]. As far as the class maintainability prediction is concerned, a significant problem is still unidentified and unsolved. This problem is the imbalanced data problem (IDP) for class maintainability prediction. The development of a prediction model for any classification task requires dataset for training. For instance, the effective training of the prediction model for binary classification, it is necessary that the dataset contains the appropriate quantity of the cases of both of the classes. Since the class maintainability prediction is regarded as a binary classification problem in this study, therefore it is essential that for effective training of the model, the dataset consists of low maintainability and high maintainability classes. As discussed earlier, a class with low maintainability requires more revisions in the maintenance period, and the prediction of such classes in the earlier phases is the prime concern of the software developers. However, in most of the maintainability prediction datasets, the classes with low maintainability are rare. If in a maintainability prediction dataset, the data points of low maintainability class are few, it is challenging to train the prediction models so as the predict the unseen data points of both classes (i.e., a class with low maintainability and high maintainability) with reasonable accuracy. In the IDP for our case, the class with low maintainability is regarded as minority classes while the class with high maintainability is the majority class.

Many real-world applications like DNA identification [16], detection of fraud transactions [17], detection of fake websites [18], etc. encounter the IDP. In the software engineering domain, the IDP is encountered in predicting defective [19,20] and change prone classes [21]. Various solutions are also proposed by the researchers to cope with this problem. A recent survey published by Haixaing et al. [22] given two primary strategies to deal with the IDP, namely data pre-processing and cost-sensitive classification. The data pre-processing techniques to cope with the IDP are data resampling techniques that tend to alter the distribution of points of the minority and majority class. The cost-sensitive techniques are generally algorithmic level solutions to IDP that work by assigning a high cost to misclassified data points. The IDP in class maintainability prediction is still unexplored. This study deals with IDP for class maintainability prediction by data resampling techniques ranging from 9 oversampling and three undersampling methods. Before developing the class maintainability prediction models, the imbalanced datasets are treated with the application of data resampling techniques, and then the balanced dataset is passed to the classification techniques. In the literature, there have been multiple studies using ML and statistical techniques to develop class maintainability prediction models [12–15]. From the last few years, in addition to ML techniques, SB techniques are gaining popularity in predictive modeling. In software engineering domain SB techniques have been applied to develop effective prediction models to predict the defective and class prone classes [23,24]. SB techniques search the entire search solution space and give an optimal or nearly optimal solution for the problem in hand. The prediction models developed using SB techniques have delivered better performance than that of ML techniques for various prediction problems. There are very few studies that have applied SB techniques for class maintainability prediction reported in [25]. Therefore, this study deals with evaluating the predictive capability of SB techniques. Apart from this, the study also extensively compares the predictive performance of various ML techniques for class maintainability prediction. The imbalanced data sets before the applications of the above-discussed methods are balanced by applying twelve data resampling techniques. The contributions of this paper are listed below.

The use of various data resampling techniques to pre-process the imbalanced datasets. The study extensively compares and evaluates different data resampling techniques to balance the imbalanced datasets before learning the models.

The use of multiple learning techniques and sound comparison between them: The study develops class maintainability prediction models by applying 14 ML and 14 SB techniques. The study extensively compares the performance of developed class maintainability prediction models with an aim to guide the software practitioners to select the best model.

Use of several executions SB techniques: The SB techniques are non-deterministic in nature. To deal with this and provide the unbiased predictions, we carry out several executions of these techniques.

Use of proper performance metrics to evaluate the predictive capability of the developed models and their statistical analysis: If the prediction models developed from the imbalanced datasets are assessed with the help of traditional performance metrics like accuracy, the results are generally biased towards the majority class. Therefore, keeping this point in mind, the paper uses the performance metrics, g-mean (GM), and balance (BL). These metrics are recommended in the domain of IDP. Also, the paper conduct statistical analysis of developed prediction models to assure unbiased predictions.

The objectives of this paper are stated in the form of the following research questions (RQs);

RQ1: What is the predictive performance of class maintainability prediction models developed on imbalanced data?
RQ2: Does the performance of class maintainability prediction models improve after employing data resampling method and to what extent the performance of the models improve?

RQ3: Which data resampling method best improves the performance of the class maintainability prediction models?
RQ4: What is the comparative performance of ML and SB techniques after data resampling for developing class maintainability prediction models?

The study is organized as follows: Section 2 describes the related work and Section 3 explains various elements of the study. Section 4 provides of the experimental layout and Section 5 explains the research methodology. The obtained results are presented and analyzed in Section 6. The threats to empirical validity are discussed in Section 7 and finally, conclusions of the study are given in Section 8.

2. Related work

The related work presented in this paper is two-fold. First, it describes the studies that have built prediction models to forecast software maintainability, and afterward, this section describes the studies taking care of the IDP.

The software practitioners and researchers have realized the importance of the prediction of maintainability of classes or modules in the early software development stages. Hence in the past, numerous studies have been published. The maintainability prediction models have been built with the help of various kinds of statistical and ML techniques by using software metrics quantifying different software characteristics. Li and Henry [5] suggested that the OO metrics are best predictors of maintainability of the OO systems. The regression-based model has been used in this study to estimate the software maintainability. Khoshgoftaar and Szabo [26] applied neural network modeling with principal component analysis to improve the accuracy of prediction models for software maintainability. The study used procedural metrics extracted from the software written in assembly language to train the neural network. Dagpinar and Jahnke [11] investigated the relation of a wide range of OO metrics with the class maintainability. The study employed multivariate regression with stepwise selection for picking up the parameters that have a strong correlation with class maintainability to build the prediction models. The study by Quah and Thwin [27] predicted class maintainability with the help of training two different kinds of neural networks, namely ward network and general regression neural network. The dataset for training the said networks was extracted from two commercial software systems. Koten and Gray [27] trained Bayesian network by using OO metrics for predicting class maintainability. The study advocated that the Bayesian model is better as compared to the regression-based models. Zhou and Leung [12] examined modeling techniques based on multiple adaptive regression splines to develop class maintainability prediction models. They compared the performance of their proposed method with the multivariate regression model and found multiple adaptive regression splines better for predicting class maintainability. A novel TreeNet model was investigated by Elish and Elish [28] to develop a prediction model for class maintainability. The accuracy of TreeNet was found superior to that of models developed in [12]. Later Jin et al. [13] developed class maintainability prediction models by project pursuit regression with OO metrics as predictors, and the performance of their models was found better than that of [12]. The study published by Olatunji [29] employed an extreme learning machine and sensitivity based methods for linear learning to predict the class maintainability. Zhang et al. [30] developed maintainability prediction models using twenty-four ML techniques. In this study, software metrics were extracted at four different levels to train the prediction models. Kumar et al. [31] examined the effectiveness of OO metrics for the prediction of class maintainability. The study used Neural-Genetic algorithm in which OO metrics are optimized using genetic algorithm before learning the neural network. Alsolai and Roapr [32] conducted a systematic review on software maintainability prediction and evaluated the performance of models developed using ML techniques w.r.t. mean magnitude of relative error (MMRE) as software maintainability is measured as a continuous variable in the form of number of changes made in the source code during the maintenance. A study by Al Dallal [8] develop prediction models to locate class with low maintainability using OO metrics. This study used three dataset namely, Illusion, FreeMind, and JabRef to develop models for predicting classes with low maintainability like the present study. All three datasets, used for developing the class maintainability prediction models in are highly imbalanced. Although the study by Jehad Al Dallal [8] developed classification models for predicting low maintainability classes but they have not addressed the issue of imbalanced data. Therefore, in this way, we see that the prediction of class maintainability in early software development phases is widely acknowledged in the past. But the problem of IDP has not even touched in this area. The IDP has encountered in many real-world applications like the prediction of cancer gene expressions [33], detection of fraudulent credit card transactions and telecommunications [17] to name a few. In this section, we will be discussing the IDP from the perspective of software quality predictive modeling (SQPM). The defective and the change prone class are the rare events that are likely to be predicted with reasonable accuracy. In SQPM, there have been multiple studies that have proposed various kind of solutions to tackle IDP. Choeikiwong and Vateekul [34] introduced a threshold adjustment strategy to support vector machine (SVM) and proposed the unbiased version called R-SVM to solve IDP in software fault prediction. The performance of R-SVM is compared with various traditioned ML techniques, and R-SVM was found to be far superior to handle IDP. Khoshgoftaar et al. [35] interrogated four different scenarios of feature selection along with data resampling to handle IDP while predicting the faulty classes. The study concluded that selecting the right set of features from the resampled data improves the performance of the prediction models. Laradji et al. [36] proposed an average probability-based ensemble (APE) incorporating seven base classifiers to handle IDP in software defect prediction. The study concluded that feature selection combined with ensemble learning gave the enhanced performance of prediction models developed from imbalanced defect datasets. Siers and Islam [37] proposed an ensemble of decision tree to tackle IDP. The cost-sensitive voting scheme has been employed in this study to minimize the classification cost. The study confirmed the superiority of the proposed technique over a few existing cost-sensitive learning methods. Pelayo and Dick [38] examined the impact of synthetic minority oversampling technique for building defect prediction models using imbalanced datasets. The study observed a much improved predictive performance with the use of SMOTE. Menzies et al. [39] pointed out that the methodologies like data resampling and boosting increase the performance of defect prediction models developed from imbalanced datasets. Seliya et al. [40] proposed a roughly balanced bagging (RRB) algorithm to deal with imbalanced datasets in software defect prediction. Their proposed RRB algorithm combined data resampling with bagging to deal with imbalanced data. The study concluded that their proposed RRB is a better choice for effectively developing software defect prediction models as compared to the conventional ML classifiers. A study by Seiffert et al. [41] comprehensively analyzed various data resampling methods along with bagging to enhance the performance of decision tree classifier learned from imbalanced data for predicting defective classes. Thus, in this way, we observe the in SQPM, the IDP is primarily being addressed in software defect prediction. This paper comprehensively investigates twelve data resampling methods, including nine oversampling and three undersampling methods considering the seriousness of the IDP for class maintainability prediction. Also, we effectively analyze the performance of fourteen ML and fourteen SB techniques to develop class maintainability prediction models. The predictive performance of the developed models is assessed with the help of performance metrics GM and BL.

3. Elements of the empirical study

In this section we describe various elements required to carry out this research.

3.1. Systems investigated

The software systems validated in this study are eight Apache application software packages that are downloaded from the GitHub repository. Class maintainability prediction models are developed using datasets extracted out of these systems. The systems investigated in this study are Bcel, Betwixt, Io, Ivy, Jcs, Lang, Log4j, and Ode. The brief description of the systems are as follows: Bcel is developed to provide a user-friendly way to create, modify, and analyze the Java class files. In case there is something wrong in the code, it gives better information to the users as compare to JVM (Java virtual machine) messages. Betwixt library is developed to transform the Java beans to XML. It also generates the diesters rules can be customized according to per type manner just like BeanInfo mechanism. Apache IO is a library that contains various utilities related to input and output. Ivy is the dominant dependency management system. It is integrated with Java build management to manage multiple kinds of project dependencies. Jcs (Java caching system) speeds up the applications by managing the cache data. In addition to cache management like remote synchronization, recovery management and remote store, to name a few. Log4j is a logging utility that makes the logging process easy and standardized. It is divided into three components logger, formatter and handler. The logger component captures the messages that are to be logged and passes to the formatter that formats the messages for the output. Lang provides numerous utilities that standard java libraries unable to provide. The main utilities provided by the Lang include string manipulation, object reflector, and concurrency control mechanisms. Ode (Orchestration Director Engine) is developed to execute the business processes written in BPEL (Business Process Execution Language). It communicates with the web server by sending and receiving messages. Apart from communicating with the webserver, it also includes various types of data manipulations and error recovery mechanisms.

3.2. Variable used in the study

The problem of predicting class maintainability has been based on the model that is developed using the set of variables or OO metrics describing and measuring a different aspect of a class like cohesion, coupling, complexity, inheritance depth, etc. In the context of developing a prediction model, these variable are regarded as independent or input variables. The class maintainability is estimated by establishing a relationship with the independent variables. The independent variables used for model construction in this study are 18 OO metrics that measure different characteristics of a class [42].

The output or the dependent variable in this study is class maintainability having two outcomes: low maintainability and high maintainability. The prime objective of this study is to estimate the low maintainability classes (challenging to modify) before releasing the software. A summary of the variables used in the study is given in Table 1.

Table 1
Variables used in the Study.

Variable	Description
Independent variables	
rfc (response for class)	Estimates the number of methods invoked on receipt of message by object of a given class.
dit (depth of inheritance tree)	Length of longest path from the root to a given class in the inheritance hierarchy.
noc (number of children)	Counts the number of immediate subclasses of a particular class.
wmc (weighed methods per class)	Aggregate of the complexities of methods in a given class.
lcom (lack of cohesion in methods)	Computed on the basis of the number of method pair in a given class that does not share any of the attributes.
cbo (coupling between objects)	Number of methods in a class accessing the methods of other classes.
mfa (measure of functional abstraction)	Sum of the number of methods inherited by a given class and the number of methods accessed by the methods of that class.
moa (measure of aggregation)	Total number of derived data types defined in a class.
cam (cohesion among the methods of a class)	Measure of cohesion among the methods of a particular class on the basis of their parameter list.
dam (data access metrics)	Ratio of the sum of private, protected datatypes in a class to that of the total number of datatypes defined in that class.
npm (number of public methods)	The total count of public attributes defined in a class.
ca (afferent coupling)	Number of classes using a particular class.
ce (efferent coupling)	Number of classes used by a particular class.
cbm (coupling among methods of a class)	Estimates the number of methods in a given class by which the inherited methods are coupled.
amc (average method complexity)	Average size of the method of a class.
lcom3 (lack of cohesion)	Estimates the cohesiveness in the methods of a given class on the basis of sharing of attributes.
ic (inheritance coupling)	Calculates the number of ancestor classes to which particular class is depending.
sloc (lines of code)	Total number of source code lines in a particular class.
Dependent variable	
CM (Class Maintainability)	Describes the class maintainability as low or high.

3.3. Evaluation metrics

The traditional way to evaluate the performance of the classifier is Accuracy and Error-rate. However, to assess the models developed over the imbalanced domain, these metrics have no sense. The accuracy metric is highly biased towards the majority class, neglecting the correct predictions for the minority class. Therefore, in this study, we evaluate the prediction models by GM and BL. The use of these metrics is advocated in the imbalanced domain. The evaluation metrics GM and BL are derived from the contingency matrix given in Table 2.

Class maintainability prediction is addressed as a binary classification problem in this paper. The low maintainability class is the minority (positive) class whereas the high maintainability class is the majority (negative) class. The evaluation metrics GM and BL are calculated as follows:

Table 2
Contingency Matrix.

	Predicted Class	
Real class	tp (true positives)	fn (false negatives)
	fp (false positives)	tn (true negatives)

$$\text{True positive Rate (T.P.R)} = \frac{tp}{tp + fn}$$

$$\text{True negative rate (T.N.R)} = \frac{tn}{tn + fp}$$

$$\text{False positive rate (F.P.R)} = \frac{fp}{fp + tn}$$

$GM = \sqrt{T.P.R * T.N.R}$; GM is the geometric mean of the true positive rate (sensitivity) and true negative rate (specificity). Unlike accuracy, this metric makes a perfect balance between the accuracies of minority as well as majority class.

$BL = 1\sqrt{\frac{(0-F.P.R)^2+(1-T.P.R)^2}{2}}$; BL is another robust and stable evaluation metric that measure the Euclidian distance between the pair (T.P.R, F.P.R) to a optimal value of the pair (T.P.R, F.P.R). In the optimal value of pair (T.P.R, F.P.R), T.P.R = 1 and F.P.R = 0.

3.4. Statistical analysis

To provide statistical support to the results of the study and strengthen the conclusions, we use statistical tests. Mainly, we use non-parametric tests to evaluate the hypothesis framed in the study. We use two types of tests (i) Friedman test (ii) Wilcoxon signed-rank test.

Friedman test [43]: This test is used to investigate the difference in the performance among multiple techniques over multiple datasets with a null hypothesis stating that all the techniques give the same performance. For each dataset, the Friedman test computes the rank of each method. The lower the grade obtained by the technique, better will be that technique. Finally, the mean rank of a technique is calculated by taking the average of rank received over all the datasets.

Wilcoxon signed-rank test [43]: We use the Wilcoxon signed-rank test for post-hoc analysis. This test is applied to investigate the existence of a significant difference between the pair of techniques. The post-hoc analysis enables us to examine whether a null hypothesis stating that there is no significant difference in the performance of a couple of techniques could be rejected or not. There p-value (lowest level of significance associated with each pair of technique) associated with each pair of technique is computed, and a decision regarding the acceptance and rejection of null hypothesis is made. In this study, both of the tests are applied at a level of significance α = 0.05.

4. Experimental layout

This section demonstrates the data acquisition procedure, pre-processing carried on datasets, model construction and validation procedure.

4.1. Maintenance data acquisition procedure

This step aims at the collection of the dataset from the software systems described in Section 3.1. The data collection is the core process in developing class maintainability prediction models. To extract data from the eight software system described in Section 3.1, we used data collection and reporting (DCRS) software [44]. This software has been successfully used in many empirical studies for data extraction out of many open source repositories that use Git version control for tracking the changes in the software. The software systems investigated in this study are different application packages of Apache which use the Git version control system; hence, the prerequisites using DCRS software were met for our research. The DCRS software analyses the two versions of a software fed as input to it and extracts the log records about common classes between the two versions. Each log record contains the following information: description of changes made, number of lines of source code added ($SLOC^{added}$), number of lines of source code deleted ($SLOC^{deleted}$), number of lines of source code modified ($SLOC^{modified}$), total number of changes ($SLOC^{total}$) where $SLOC^{total}$ = Sum($SLOC^{deleted}$, $SLOC^{added}$, $SLOC^{modified}$). Each log record also contains a variable Alter containing "yes" and "no" values. If the value of the Alter variable is "yes", this means that the class is changed during the maintenance otherwise it is not changed. The CKJM tool that functions as a OO metric calculator is embedded within DCRS to extract OO metrics corresponding to the software systems to be analyzed. Thus, DCRS software generates datapoint corresponding common classes on analyzing two given versions of a software system. Each generated data point will be of the form <X_i, Y_i> where X_i = <rfc_i, dit_i, noc_i, wmc_i, $lcom_i$, cbo_i, mfa_i, moa_i, cam_i, dam_i, npm_i, ca_i, ce_i, cb_i, amc_i, $lcom3_i$, ic_i, loc_i, $SLOC_i^{added}$, $SLOC_i^{deleted}$ $SLOC_i^{modified}$, $Alter_i$> here rfc_i, dit_i, noc_i, wmc_i, $lcom_i$, cbo_i, mfa_i, moa_i, cam_i, dam_i, npm_i, ca_i, ce_i, cb_i, amc_i, $lcom3_i$, ic_i, loc_i are OO metrics descrided in Table 1, $SLOC_i^{added}$ is the number of lines of code added in ith common class, $SLOC_i^{deleted}$ is number of lines deleted in ith common class, $SLOC_i^{modified}$ is number of lines of code modified in ith common class, $Alter_i$ is the value of the Alter variable for the ith common class and Y_i = $SLOC_i^{total}$. The equality Y_i = $SLOC_i^{total}$ signifies the class maintainability as a function of the number of revised lines of code in a class during the maintenance phase. In this study, the dependent variable CM is formed by dividing Y_i into two bins based on the volume of the revised lines of code. Therefore, in this way, the set of data points <X_i, Y_i> form dataset corresponding to a software system.

4.2. Data pre-processing

This section describes various pre-processing steps carried out on the extracted datasets.

(i) *Exclusion of datapoints from the dataset in for which the value of the Alter variable was {no}:* First of all, we removed the data points from the datasets in which the value of the Alter variable was no. The "no" value of the Alter variable signifies that the corresponding class is not involved in maintenance (in terms of revisions in lines of code). Table 3 summarizes the characteristics of the eight extracted datasets, including the versions analyzed, the number of common classes (#Cls.), the number of the common class changed (#Changed Cls). For example, the table depicts that for Bcel dataset, there are 263 classes that common between the two analyzed versions and 260 classes are changed (Alter variable was "yes") during the maintenance.

(ii) *Detection and removal of outliers:* In this step, we analyzed data points that were having extreme variability from the remaining data points in the dataset. Such data points are regarded as outliers. It is to be noted that the number of data

Table 3
Number of Common Classes and Common Classes Changed for Datasets.

Dataset	Version	#Cls	(#Changed Cls)
Bcel	5.0–5.1	363	360
Betwixt	0.6–0.7	290	279
IO	2.0.1–2.2	274	272
Ivy	1.4.1–2.2.0	619	429
Jcs	1.2.6.5–1.2.7.9	333	219
Lang	2.4–2.6	434	267
Log4j	1.2.14–1.2.15	491	437
Ode	1.3.1–1.3.2	1060	1004

points in the datasets used in the study is equal to the number of common classes changed. For example, as depicted in Table 3, Bcel dataset contains 360 data points, and those 360 data points were considered for outlier analysis for Bcel dataset. The removal of outliers is critical as these points have a negative impact during the model training. In this study, we used the Interquartile filter available in the Waikato Environment for Knowledge Analysis(WEKA) tool [45]. Table 4 describes the number of outliers detected (#Outl.) and the number of data points after outliers removal (#datapts.) for each of the dataset used in the study.

(iii) *Selection of relevant features:* In a predictive modeling task, the selection of appropriate features has a substantial influence on the performance of the learned model. The feature selection process eliminates the irrelevant and redundant features from the training dataset. The advantages of feature selection can incorporate a decrease in the information expected to accomplish learning, reduction in execution time, and enhanced prediction accuracy of the model. In this study, correlation-based feature selection (CFS) [46] has been used to select the relevant features in the training datasets. CFS selects those features that have a high correlation with the dependent variable and least correlation among themselves. A study by Hall and Holmes [47] assessed six different feature selection techniques on fifteen datasets. This study discovered that although there is no excellent technique for feature selection, the results of the CFS technique were promising. A comprehensive review of 64 fault prediction studies in the time of 1991 to 2013 was conducted by [48]. This study revealed that CFS was one of the most commonly used feature selection technique. Also, in predictive modeling in software engineering, CFS technique has successfully applied in various studies [49–51] to develop prediction models. Hence, we use this feature selection technique in this study. The CFS results are shown in Table 5.

(iv) *Data discretization*: In this empirical study, the dependent variable CM is formed by discretizing the continuous variable Y_i into two bins and the assigning labels LMC (low maintainability class) and HMC (high maintainability class) for each data point. As explained in Section 4.1, Y_i signifies the class maintainability as a function of the number of SLOC revised in the maintenance period. On closely looking at the Y_i for all datasets used in the study, we found that Y_i values ranged from very few SLOC revised to thousands of revisions in SLOC. The values of the dependent variable CM for a particular data point in our analysis are assigned on the basis of continuous variable Y_i as follows: If the value of Y_i for a data point d_i is greater than average Y_i values for all data points in the dataset, CM value was set to LMC; otherwise the value of CM was set to HMC. The software practitioners are advised to critically test and well document the classes that are predicted to be LMC. Table 6 presents the number and proportion of data points for each value of our discretized dependent variable.

Table 4
Results of Outlier Analysis.

Dataset	#Outl.	#datapts
Bcel	26	334
Betwixt	34	245
IO	24	248
Ivy	58	371
Jcs	22	197
Lang	47	220
Log4j	32	405
Ode	115	889

Table 5
Selected Features using CFS.

Dataset	Selected features
Bcel	wmc, cbo
Betwixt	wmc, cbo, rfc, lcom, sloc, cam
IO	npm, lcom3
Ivy	wmc, npm, sloc, cam
Jcs	wmc, lcom3, sloc, cam
Lang	npm lcom3, sloc
Log4j	npm lcom3, sloc, dam, cam
Ode	wmc, cbo, rfc, lcom, sloc, cam, moa, ca, ce

Table 6
Number and Proportion of Datapoints for each value of Discretized Dependent Variable.

Dataset	CM	No. of Datapoints & Percentage
Bcel	LMC	18 (5.38%)
	HMC	316 (94.61%)
Betwixt	LMC	27 (11.02%)
	HMC	218 (88.97%)
Io	LMC	10 (4.03%)
	HMC	238 (95.96%)
Ivy	LMC	28 (7.45%)
	HMC	343 (92.45%)
Jcs	LMC	27 (13.70%)
	HMC	170 (86.29%)
Lang	LMC	27 (12.27%)
	HMC	193 (87.72%)
Log4i	LMC	42 (10.37%)
	HMC	363 (89.62%)
Ode	LMC	57 (6.41%)
	HMC	(93.58%)

4.3. Data balancing

Table 6 describes the number and percentage of data points belonging to the LMC and HMC class. It is evident from Table 6 that there is a high imbalance in the class distribution of the data points belonging to the two classes. For instance, in Bcel dataset, there are 5.38% datapoints of LMC class, and 94.61% of the data points belong to the HMC class. Similarity, for the rest of the datasets of the study, we encounter a similar situation. The LMC class is the minority class as for each dataset there is very less number of data points of this class. The HMC class is the majority class. The principal objective of this study is the prediction of LMC classes with high accuracy. If we train the prediction model with such imbalanced datasets, the HMC classes would be predicted with reasonable accuracy but LMC classes that are of our prime interest may not the predicted accurately as the predicted model would not have learned the sufficient data points of the LMC class. Therefore, it is necessary to balance the class distribution of the data points of the two classes. To do so, we apply various data balancing techniques and study the behaviour of the prediction models after data balancing. The brief description of the data balancing techniques used in this study is given in the next section.

4.4. Development and evaluation of class maintainability prediction model

The next step in our experimental setup is developing class maintainability prediction models. We apply various ML and SB techniques and examine their effectiveness for the problem in hand. The forthcoming section gives a brief description of all the techniques. During learning the model, we apply ten-fold cross-validation. The ten-fold cross-validation mechanism works by dividing the training dataset into ten equal-sized partitions. At random, nine partitions are utilized for learning the model, and the performance of the learned model is estimated with the leftover tenth partition. This process is repeated such that every partition is used at least once for testing the performance of the learned model. The various performance metrics on which the performance of class maintainability prediction models is evaluated are discussed in Section 3.3.

5. Research method

5.1. Data resampling techniques

This section defines the data resampling techniques used in the study to balance the distribution of the LMC and HMC datapoints. In the context of our research, LMC refers to minority class, and HMC refers to the majority class. The data resampling techniques used in the study are of two types: oversampling and undersampling. The oversampling techniques balance the data points of the two classes by reproducing the data points of the minority class [52]. The undersampling techniques randomly throw-out the data points of the majority class to balance the data distribution [52[. Table 7 describes the data resampling techniques used in this study in brief.

5.2. Classification techniques

This section describes the classification techniques used in the study to develop class maintainability prediction models. We extensively compare and assess the performance of 28 methods for predicting HMC classes. Out of 28 techniques investigated in this study, 14 are ML techniques, and 14 are SB techniques. Except for a few ML techniques, remaining methods have not examined in any of the study in literature for class maintainability prediction. A summary of the classification techniques is given in Table 8. For developing class maintainability prediction using classification techniques described in Table 8, we have used Knowledge Extraction based on Evolutionary Learning (KEEL) tool [61].

In order to promote the replicability and repeatability of the results, executed these techniques with the default set of parameters. The detailed description of methods described in Table 8 can be referred from [61].

6. Results and analysis

In this section, we answer the research questions by analyzing the results of the class maintainability prediction models.

6.1. RQ1: What is the predictive performance of class maintainability prediction model developed on imbalanced data?

To answer this RQ; we developed class maintainability prediction models by using the imbalanced datasets. The ML and SB techniques described in Table 8 were used to create the models by providing the datasets using ten-fold cross-validation. The performance of developed models is assessed using GM and BL metrics (shown in Tables 9 and 10). In Tables 9 and 10, the first fourteen rows show the GM results of class maintainability prediction models developed using SB techniques and the next fourteen rows depicts the performance of models developed using the ML technique. For SB techniques, we developed models by running each technique ten times to take care of the scholastic nature of these

Table 7
Summary of Data Resampling Techniques.

Oversampling Techniques	
ROS (Random oversampling) [54]	This is an oversampling technique that produces random datapoints corresponding to the majority class such that both classes have same distribution of the datapoints.
SMOTE (Synthetic Minority Oversampling Technique) [55]	The minority class is oversampled by producing synthetic datapoints along the line joining the minority class datapoint and its k-nearer neighbours. By default, the value of k is taken as 5. If one nearer neighbour is picked up and a synthetic datapoint is generated corresponding to it, 100% oversampling of minority class results in.
BSMOTE (Borderline Synthetic Minority Oversampling Technique) [56]	Synthetic datapoints are created only for those minority class datapoints that are at the borderline. A minority class datapoint known be a borderline datapoint if all of its k nearer-neighbours are belongs to majority class.
Adasyn (Adaptive Synthetic Sampling) [53]	Uses the density distribution of minority for each minority class datapoint to decide the rate of oversampling. The synthetic datapoints corresponding to a minority class datapoint are generated in accordance to its level of difficulty in learning.
Safe-SMOTE (Safe level synthetic minority oversampling technique) [57]	Contrasting with SMOTE, Safe-SMOTE estimates the safe level of each minority class datapoints prior to creating synthetic datapoints corresponding to them. The safe level of a minority datapoint is the number of datapoints belonging to minority class in the K-nearer neighbours. If the value of safe-level is close to zero, the datapoint is thought off noise and if this value is close to k then the datapoint is thought off safe.
SMOTE-TL (synthetic minority oversampling technique with Tomek's link) [54]	Synthetic datapoints are created using SMOTE and then Tomek's link are recognized and rejected from the dataset. Tomek's links are two datapoints such that one belongs to minority class and the other belongs to majority class and the distance between them is less than distance between those datapoints to any other datapoint in the dataset.
SMOTE-ENN (synthetic minority oversampling technique with edited nearer neighbours) [54]	Synthetic datapoints are created using SMOTE and noisy datapoints are eliminated from the dataset by Wilson's ENN rule.
SD1 (Selective pre-processing of imbalanced data) [58] & SD2 (Selective pre-processing of imbalanced data-II) [59]	This method involves oversampling the minority class instances and stifling the datapoints of the majority class on the basis of whether they are nosy or safe. Then the noisy datapoints are discarded. Unlike SD1, in SD2, the noisy datapoints of majority class are either discarded or relabelled depending upon relabel option.
Undersampling Techniques	
RUS (random undersampling) [54]	In this technique the strength of majority class is decreased by arbitrarily removing the datapoints.
CNN (Condensed nearer neighbours) [53]	Unlike randomly removing the majority class datapoints, it removes the majority class datapoints in such a way the performance of near neighbour classification is not affected.
NCL (Neighbourhood Cleaning Rule) [60]	The majority class is undersampled using edited nearer neighbour rule (ENN). If the class label of a majority class datapoint is different than that of three of the five nearer neighbours, that datapoint is removed.

Table 8
A Brief Summary of Classification Techniques.

Classification technique	Description
ML Techniques	
AdaBoost (AB), Bagging BG)	AB and BG are ensemble techniques. AB develops a powerful classifier by improving up the performance of several weak classifiers. BG works by partitioning the training dataset into several subsets and developing classifier form each subset. It finally combines the predictions of individual models using majority voting. Both of these techniques used C4.5 decision tree as the component classifier.
k-Nearer Neighbour (*k*-NN), KStar (KS)	*k*-NN & KS are instance based classification techniques. *k*-NN classifies an instance on the basis of majority voting on its *k* neighbouring instances. KS uses entropy as measure of dissimilarity.
Logistic Regression with ridge estimator (LRR)	LRR combines ridge estimator with logistic regression to solve the problem of multicollinearity.
C4.5, Pruning and Building Integrated in classification (PUBLIC), Partial Decision Trees (PART)	C4.5 is a classification technique based on decision tree that use information gain to decide splits on nodes. PUBIC is a decision tree classier formed by integrating tree building and pruning as one approach. PART develops the partial decision tree using C4.5 and then prunes it to obtain final decision tree.
Repeated Incremental Pruning to Produce Error reduction (RIPPER), Simple Learner with Iterative Pruning to Produce Error Reduction (SLIPPER)	RIPPER is a rule based classifier that first develops the rule set from the training dataset and the continuously improves it. SLIPPER is a boosting algorithm in which the weak rules are re-weighted and united using ensemble to make strong rules.
CHIRW (Weighted Rule based Learning model given by Chi et al)	This is a rule based classification technique that predict the label of datapoint using set of fuzzy weighted rules.
Batch Nested generalized Exemplar (BNGE), Rule Induction from set of Exemplars (RISE), Exemplar-aided Construction of Hyperrectangles (EACH)	BNGE, RISE and EACH are in-memory algorithms that determine the label of the test example on the basis of distance measure between the test example and the examples kept in the memory.
SB Techniques	
Genetic Algorithm based classification system with Adaptive Discretization Intervals (GAssist-ADI), Genetic Algorithm based classification system with Intervalar Rules (GAssist-Int)	GAssist-ADI consists of classification using rule set called decision list Further it uses Genetic Algorithm (GA) to select the right discretization of each generated rule. GAssist-Int is GA based classification techniques that uses Intervalar for representing classification rules.
X-classification system (XCS), supervised classification system (UCS)	This is a reinforcement learning based classification techniques using accuracy fitness. UCS is a supervised learning based classification system.
Bioinformatics based hierarchical evolutionary learning (BIOHEL), Memetic Pittsburgh learning classification system (MPLCS)	This technique divides the training examples into non-overlapping subsets and uses GA for rule generation. MPLCS is rule based classification systems integrated with rule-wise local search to improve convergence.
Constricted particle swarm optimization (CPSO), Linear decreasing weight particle swarm optimization (LDWPSO)	CPSO is the variant of PSO that uses constricted factor to avoid pre-mature convergence to local maxima. LDWPSO is another variant of PSO that uses linear deceasing weight to avoid pre-mature convergence to local maxima.
Generational genetic algorithm for instance selection (GGA), Steady state algorithm for instance selection (SGA), CHC adaptive search for instance selection (CHC), Population based incremental learning (PBIL)	GGA, SGA, CHC & PBIL are instance selection techniques based on evolutionary computing. In GGA population of chromosomes representing training datapoints is continuously evolved through controlled variations. SGA selects two parents from the population, create offstring using crossover and mutations and then evaluates the offstring with appropriate fitness function. CHC uses special form of recombination operator in the recombination phase. PBIL technique keep information of the entire search space in order to take decision to sample next.
Intelligent genetic algorithm for edition (IGA), Steady-state Memetic algorithm for instance selection (SSMA)	Rather than conventional crossover, IGA make use of intelligent crossover for parameter optimization. SSMA combines SGA with local search where SGA picks up good starting datapoints and local search offers precise representation of the search region.

techniques and reported the median values for the performance metrics GM and BL.

For each of 14 ML techniques on 8 datasets, 14 × 8 = 112 models were developed. For each of 14 SB techniques on 8 datasets by running each SB techniques 10 times, 14 × 8 × 10 = 1120 models were developed. Thus, in this experiment, we assessed the performance of 1232 (112 + 1120) class maintainability prediction models.

On analyzing the GM results (Table 9) it was observed that GM values are less than 50 in 42.85% of the cases for Bcel dataset, 85.71% of the cases for Betwixt dataset, 53.57% of the cases for Io dataset, 92.85% of the cases for Ivy dataset, 17.85% of the cases for Jcs dataset 7.14% of the cases for Lang dataset, 57.14% of the cases for Log4j dataset and 85.71% of the cases of Ode dataset. Similarly, the performance analysis on BL results (Table 10) it was discovered that, BL values were less than 50 in 50% of the cases for Bcel dataset, 89.28% of the cases for Betwixt dataset, 53.57% of the cases for Io dataset, 92.85% of the cases for Ivy dataset, 25% of the cases for Jcs dataset 10.72% of the cases for Lang dataset, 51.14% of the cases for Log4j dataset and 100% of the cases of Ode dataset. The median of GM and BL of the class maintainability prediction models developed on imbalanced datasets are reported in Figs. 1 and 2.

As shown in Fig. 1 median GM values for all datasets except Jcs and Lang are very low. For Bcel dataset median GM is reported to be 54.69. Similarly, median BL values for all datasets except Jcs and Lang were very much low. For Bcel dataset the median BL was just close to 50. Therefore, it can be seen that the performance of class maintainability prediction models developed from the imbalanced dataset is very poor. The reason of the poor performance is the fact that the datasets have very few instances of the LMC due to which the models have not competently been able to learn the instances of LMC classes and resulted in very low T.P.R.

6.2. RQ2: Does the performance of class maintainability prediction models improve after employing the data resampling methods and to what extent the performance of the models improve?

In our second experiment, we develop class maintainability prediction models by applying data resampling techniques on the imbalanced datasets. We developed the prediction models using

Table 9
GM Results of Class Maintainability Prediction Models on Imbalanced Datasets.

Technique	Bcel	Betwixt	Io	Ivy	Jcs	Lang	Log4j	Ode
BIOHEL	61.67	35.74	46.54	40.50	67.11	76.42	58.67	44.25
CHC	52.29	0.00	31.62	18.87	63.10	79.07	21.61	37.37
CPSO	70.69	52.75	73.11	44.36	57.92	79.31	53.21	39.95
GGA	46.84	0.00	40.74	18.79	77.22	79.29	45.78	41.66
GAssist-Int	66.14	0.00	54.66	26.69	74.91	67.94	58.68	29.58
GAssist-ADI	46.84	26.78	0.00	37.41	65.28	75.37	54.94	37.42
IGA	46.69	40.70	43.96	31.76	68.54	60.26	44.67	48.51
LWPSO	67.54	56.37	77.08	31.91	40.69	77.63	44.67	51.85
MPLCS	57.18	33.03	44.53	32.68	69.86	77.90	57.18	32.41
PBIL	57.09	0.00	31.62	18.84	79.95	79.51	40.54	37.42
SGA	61.67	18.93	31.62	33.09	63.86	79.29	44.48	34.92
SSMA	57.09	18.93	31.49	26.98	52.31	65.27	54.55	29.79
UCS	40.76	19.11	54.66	32.49	52.48	65.45	56.69	41.63
XCS	23.57	0.00	44.63	26.73	42.65	60.22	27.23	26.47
AB	57.18	75.11	62.44	49.19	64.07	70.12	58.43	48.57
BG	57.28	0.00	0.00	37.58	65.68	79.72	50.54	41.63
C4.5	33.17	0.00	0.00	32.64	62.50	76.84	30.69	41.73
PART	0.00	0.00	31.42	0.00	26.98	50.25	37.37	0.00
RIPPER	58.82	60.41	70.65	56.63	78.80	76.63	65.56	51.33
SLIPPER	66.14	36.96	54.43	41.51	71.63	65.45	61.93	48.54
CHI-RW	0.00	0.00	0.00	26.73	0.00	39.12	22.06	19.38
KNN	52.29	44.12	54.54	60.45	65.61	72.78	32.85	50.05
KSTAR	46.92	0.00	54.66	26.61	58.30	62.49	46.10	0.00
LRR	33.23	33.18	54.66	46.09	67.88	57.59	34.31	27.47
PUBLIC	23.53	0.00	0.00	0.00	39.85	74.34	0.00	29.62
BNGE	38.91	41.73	54.19	47.18	64.27	67.01	60.00	47.18
EACH	60.70	32.79	74.75	32.68	80.71	27.15	48.57	32.11
RISE	57.09	51.08	62.44	47.97	64.96	58.94	26.76	52.02

Table 10
BL Results of Class Maintainability Prediction Models on Imbalanced Datasets.

Technique	Bcel	Betwixt	Io	Ivy	Jcs	Lang	Lo4j	Ode
BIOHEL	56.76	38.98	44.97	41.63	63.05	73.31	55.70	43.96
CHC	48.92	29.29	36.36	31.81	63.05	76.02	32.64	39.21
CPSO	70.23	51.43	70.68	44.14	54.98	79.23	50.80	41.35
GGA	45.00	29.28	41.07	31.81	73.54	76.09	44.42	41.69
GAssist-Int	60.70	29.28	50.50	34.34	70.95	63.22	54.47	35.49
GAssist-ADI	45.00	34.49	29.29	39.37	60.61	71.04	51.14	39.21
IGA	44.99	41.90	43.38	36.73	65.31	57.40	44.23	46.57
LWPSO	65.90	54.36	76.24	36.77	41.90	77.42	44.23	49.89
MPLCS	52.84	37.13	43.43	36.87	65.70	73.69	52.84	36.73
PBIL	52.83	29.29	36.36	31.81	76.25	76.15	41.07	39.21
SGA	56.76	31.87	36.36	37.14	60.29	76.09	52.84	37.97
SSMA	52.83	31.87	36.36	34.52	49.95	60.61	51.10	35.60
UCS	41.07	31.90	50.50	36.86	49.99	60.63	52.79	41.69
XCS	33.22	29.29	43.43	34.34	42.37	55.45	34.59	34.25
AB	52.84	71.40	57.54	46.92	60.35	65.76	54.44	46.58
BG	52.85	29.26	29.29	39.39	60.66	76.20	47.78	41.69
C4.5	37.14	29.28	29.29	36.86	58.00	73.44	36.02	41.69
PART	29.29	29.29	36.35	29.29	34.52	47.59	39.37	29.29
RIPPER	56.10	58.73	69.32	53.97	77.46	74.97	63.28	48.95
SLIPPER	60.70	39.51	50.49	41.86	68.11	60.63	57.75	46.58
CHI-RW	29.29	29.29	29.29	34.34	29.28	40.17	32.74	31.96
KNN	48.92	44.31	50.50	56.78	62.58	68.40	37.36	47.79
KSTAR	45.00	29.29	50.50	34.34	55.10	57.99	44.44	29.29
LR	37.14	37.14	50.50	44.44	63.50	52.86	37.70	32.25
PUBLIC	33.22	29.29	29.29	29.29	40.98	70.80	29.29	35.49
BNGE	40.72	42.23	50.48	39.35	60.40	63.03	56.05	45.39
EACH	57.86	37.11	74.44	36.87	80.34	34.53	47.34	36.60
RISE	52.83	49.53	57.54	46.26	62.31	55.26	34.55	49.07

ten-fold cross-validation and evaluated the performance with the help of GM and BL metrics. On each dataset, 28 learning techniques (14 ML techniques and 14 SB techniques) in conjunction with 12 data resampling techniques are used to develop the prediction models for class maintainability. For each dataset, using 14 ML techniques over 12 data resampling techniques, 14 $\times$ 12 = 168 combinations are explored. As we have used 8 datasets in this study, therefore in case of ML techniques, 8 datasets $\times$ 168 combinations = 1344 models are assessed to formulate our results. To take care of the stochastic nature of SB techniques, we run each of these techniques ten times and reported the median values of GM and BL performance metrics.

For each dataset, using 14 SB techniques over 12 data resampling techniques and 10 runs, 14 $\times$ 12 $\times$ 10 = 1680 combinations are explored. For 8 datasets in case of SB techniques, 8 datasets $\times$ 1680 combinations = 16800 models are developed,

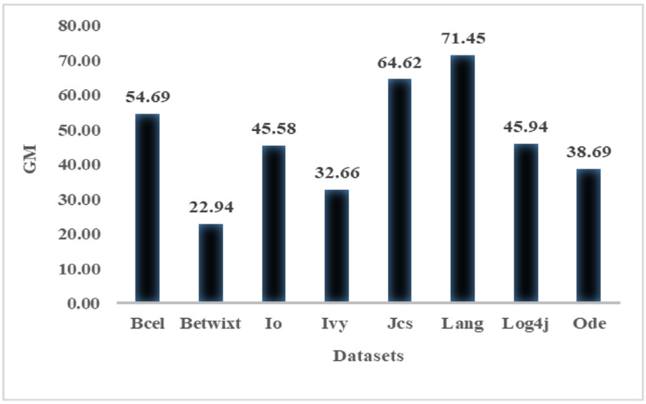

Fig. 1. Median GM values of class maintainability prediction models developed for imbalanced datasets.

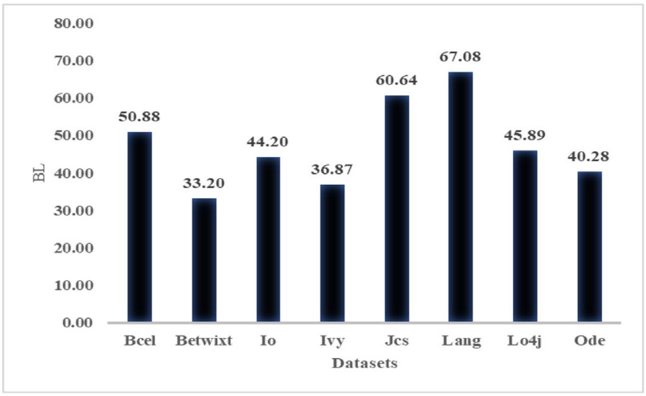

Fig. 2. Median BL values of class maintainability prediction models developed for imbalanced datasets.

and median values of GM and BL performance metrics are reported to formulate the results shown in Tables 11–26. GM results of the class maintainability prediction models developed after data resampling are presented in Tables 11–18 for Bcel, Betwixt, Io, Ivy, Jcs, Lang, Log4j, and Ode datasets. Tables 19–26 show the BL performance metric results for all eight datasets used in the study. On analyzing Tables 11–18, it was noted that GM values are higher than 50 in 88.99% of the cases for Bcel dataset, 67.26% of the cases for Betwixt dataset, 93.15% of the cases for Io dataset, 77.68% of the cases for Ivy dataset, 96.13% of the cases for Jcs dataset, 93.75% of the cases for Lang dataset, 92.55% of the cases for Log4j dataset, and 80.65% of the cases for Ode dataset. The analysis of BL results presented in Tables 19–26, for class maintainability prediction models after data resampling revealed that BL values are greater than 50 in 87.89% of the cases for Bcel dataset, 67.26% of the cases for Betwixt dataset, 92.86% of the cases for Io dataset, 71.43% of the cases for Ivy dataset, 94.34% of the cases for Jcs dataset, 91.36% of the cases for Lang dataset, 92.69% of the cases for Log4j dataset, and 78.86% of the cases for Ode dataset.

Fig. 3 shows the median of GM and BL results of class maintainability prediction models developed after employing data resampling techniques. It is evident from Fig. 3 that there is a substantial positive enhancement in the performance of prediction models after data resampling as compare to the situation when prediction models were developed from the imbalanced datasets (Figs. 1 and 2). With respect to performance metric GM, 28.07% of improvement is observed on Bcel dataset after data resampling, 155.10% for Betwixt dataset, 45.88% for Io dataset, 81.59% for Ivy dataset, 21.78% for Jcs dataset, 9.02% for Lang dataset, 46.04% for Log4j dataset, and 55.28% for Ode dataset. Also, there is a relatively significant improvement in the performance of the prediction models after data resampling for BL performance metric. The prediction models developed on Bcel dataset after data resampling gave a gain of 34% for BL. An improvement 71.89% for Betwixt dataset, 44.57% for Io dataset, 54.16% for Ivy dataset, 27.62% for Jcs dataset, 12.76% for Lang dataset, 42.28% for Log4j dataset, and 42.92% for Ode dataset is observed for BL.

Table 11
GM Results of Class Maintainability Prediction Models after Applying Data Resampling Techniques on Bcel Dataset.

Techniques	Adasyn	BSMOTE	ROS	Safe-SMOTE	SMOTE	SMOTE-ENN	SMOTE-TL	SD1	SD2	CNN	NCL	RUS
BIOHEL	73.75	75.66	72.04	78.08	71.16	69.92	69.91	68.90	68.67	69.36	75.79	75.74
CHC	71.01	75.40	70.43	79.42	73.72	69.92	73.92	69.24	72.75	72.02	76.04	74.77
CPSO	65.16	69.28	66.77	68.32	65.96	69.47	65.60	66.00	64.78	47.16	69.58	67.74
GGA	70.71	76.04	74.22	79.71	76.45	69.81	70.64	69.01	75.91	71.53	75.91	74.49
GAssist-Int	77.54	75.40	76.64	79.57	77.54	69.92	77.13	69.47	75.79	68.90	75.79	74.68
GAssist-ADI	79.16	75.27	73.76	79.16	79.16	70.04	75.79	69.92	65.19	42.54	65.82	79.42
IGA	73.92	74.89	76.62	74.24	72.53	65.92	73.53	60.56	66.68	66.56	46.16	74.77
LWPSO	65.15	69.36	68.12	67.93	66.88	65.76	66.28	68.74	75.12	54.35	67.51	60.59
MPLCS	79.16	75.79	77.54	79.16	79.16	70.04	79.16	65.71	68.78	65.28	65.82	79.85
PBIL	72.04	75.27	72.27	79.42	74.24	69.92	73.92	69.24	75.79	66.56	75.91	74.77
SGA	70.86	75.53	71.18	79.42	72.39	69.92	74.35	72.50	72.62	72.75	76.30	76.49
SSMA	66.19	74.63	65.94	81.07	77.41	61.67	48.36	73.47	75.40	72.75	76.30	76.49
UCS	40.76	40.76	40.76	40.76	40.76	40.76	40.76	40.76	40.70	33.28	33.28	33.28
XCS	67.27	72.14	63.15	63.42	55.07	69.92	58.46	57.37	61.77	52.03	69.47	70.59
AB	73.60	75.53	59.18	77.27	74.37	73.47	80.87	65.50	69.01	56.24	76.04	77.18
BG	73.29	75.27	69.11	76.72	75.61	73.47	74.35	65.39	68.32	70.79	76.17	76.04
C4.5	72.82	75.27	68.83	77.13	74.50	65.71	72.53	65.50	71.65	55.12	72.75	79.13
PART	75.27	75.27	74.63	61.47	64.74	0.00	75.05	0.00	0.00	0.00	0.00	0.00
RIPPER	0.00	33.28	0.00	23.46	0.00	73.47	52.37	55.48	57.08	53.29	56.38	80.99
SLIPPER	71.45	72.02	75.27	75.53	75.27	73.47	76.86	72.62	73.11	67.27	79.56	76.49
CHI-RW	76.55	57.37	61.87	61.67	61.67	46.99	76.17	33.23	33.23	64.00	33.23	61.57
KNN	61.87	57.28	40.50	52.29	57.28	61.97	79.95	92.80	65.17	51.32	76.04	40.95
KSTAR	71.16	72.02	55.13	75.75	70.50	69.92	72.91	69.01	68.78	47.44	65.50	75.13
LRR	77.82	0.00	77.95	80.14	80.28	70.04	76.79	69.70	73.23	33.23	69.70	76.04
PUBLIC	75.24	75.27	62.39	79.13	70.22	70.04	70.22	72.50	72.26	57.64	69.47	74.11
BNGE	68.90	68.78	63.27	68.11	66.11	73.59	66.76	62.64	64.75	35.87	68.20	75.44
EACH	60.70	60.70	60.70	60.70	60.70	60.70	60.70	60.70	60.70	60.70	60.70	60.70
RISE	67.13	72.02	55.36	79.13	70.09	69.92	84.19	72.75	72.50	55.69	72.50	57.09

Table 12
GM Results of Class Maintainability Prediction Models after Applying Data Resampling Techniques on Betwixt Dataset.

Techniques	Adasyn	BSMOTE	ROS	Safe-SMOTE	SMOTE	SMOTE-ENN	SMOTE-TL	SD1	SD2	CNN	NCL	RUS
BIOHEL	50.08	51.61	40.05	62.85	52.62	56.93	59.63	46.78	47.28	47.25	70.69	62.13
CHC	72.31	55.45	47.64	69.22	69.66	70.67	65.95	69.22	68.77	0.00	69.95	71.19
CPSO	65.43	67.58	40.58	65.90	64.07	71.03	59.90	61.46	62.17	63.58	65.07	68.40
GGA	63.86	40.28	46.80	67.94	66.04	62.21	65.03	67.22	60.41	32.79	67.92	70.51
GAssist-Int	65.24	40.49	46.09	67.73	64.49	64.71	64.73	50.82	61.91	51.22	70.69	67.98
GAssist-ADI	61.46	43.43	45.07	64.28	65.33	63.21	59.90	50.68	54.04	52.01	70.14	66.04
IGA	58.00	57.84	42.49	64.12	59.30	57.29	61.28	51.58	47.89	19.02	61.46	68.63
LWPSO	65.43	66.40	41.48	63.78	67.05	58.65	51.88	66.78	60.73	69.17	62.88	64.91
MPLCS	63.78	40.17	40.82	65.67	62.46	58.13	58.93	54.32	53.33	40.91	65.98	51.70
PBIL	62.24	55.57	44.32	69.91	64.70	67.46	66.72	61.74	58.48	19.02	63.23	68.63
SGA	64.91	47.91	42.56	67.70	67.94	62.44	65.95	56.37	61.58	42.03	60.91	72.09
SSMA	64.70	51.61	43.19	67.46	68.40	68.97	66.50	59.27	58.32	42.03	60.91	72.09
UCS	32.95	32.95	24.65	32.95	32.95	32.95	33.03	33.03	27.03	32.56	32.56	32.56
XCS	58.58	40.49	35.63	65.72	66.10	68.63	53.18	48.16	47.16	57.29	65.07	54.72
AB	58.00	53.90	32.10	59.22	47.50	57.47	63.86	50.14	46.52	26.36	71.70	61.69
BG	67.17	36.81	28.34	68.65	61.69	58.73	59.94	43.54	45.36	43.66	73.29	68.41
C4.5	60.52	44.58	31.44	45.34	61.41	55.12	64.71	39.54	44.70	32.64	68.29	66.26
PART	57.65	49.32	0.00	60.55	26.90	53.82	26.07	0.00	0.00	26.65	18.98	59.93
RIPPER	45.88	49.01	32.01	63.64	54.21	55.01	57.02	61.46	58.70	58.13	68.63	60.73
SLIPPER	57.82	54.60	41.51	67.98	62.07	58.32	64.45	50.14	46.40	52.06	70.48	57.91
CHI-RW	70.90	66.86	22.94	60.08	69.31	71.09	69.61	19.63	50.30	0.00	34.00	69.81
KNN	58.70	66.85	6.73	43.54	67.34	60.91	62.35	50.68	55.30	54.51	63.40	68.40
KSTAR	56.58	43.19	33.82	47.89	45.75	48.21	48.94	31.44	37.44	25.41	32.64	62.56
LRR	71.39	30.64	22.17	71.87	72.10	72.10	68.48	56.21	63.96	41.63	65.25	70.05
PUBLIC	68.79	51.48	28.57	62.56	68.18	62.35	64.41	57.65	62.70	0.00	41.63	16.28
BNGE	63.21	63.05	27.95	52.90	65.33	59.33	61.47	50.82	54.46	60.24	68.18	63.01
EACH	32.79	32.79	30.80	32.79	32.79	32.79	32.79	32.79	32.79	32.79	32.79	32.79
RISE	55.30	50.41	32.22	63.82	61.30	60.52	61.00	47.41	53.33	60.33	69.24	60.37

Table 13
GM Results of Class Maintainability Prediction Models after Applying Data Resampling Techniques on Io Dataset.

Technique	Adasyn	BSMOTE	ROS	Safe-SMOTE	SMOTE	SMOTE-ENN	SMOTE-TL	SD1	SD2	CNN	NCL	RUS
BIOHEL	59.69	53.73	51.34	67.36	52.54	61.08	74.13	78.96	58.27	72.76	60.94	80.34
CHC	70.47	68.60	68.95	71.01	71.18	70.83	71.18	54.43	53.26	44.34	61.63	73.56
CPSO	74.25	78.59	78.08	79.71	73.16	80.13	60.25	79.89	84.32	69.03	88.47	84.02
GGA	70.83	61.90	62.85	64.98	72.06	66.89	65.30	54.08	61.08	53.37	54.19	71.13
GAssist-Int	64.33	61.77	53.96	71.01	72.06	65.63	58.84	62.58	54.19	68.29	61.90	78.22
GAssist-ADI	63.51	68.91	69.26	76.70	70.29	64.66	71.18	54.31	53.96	66.74	60.81	74.36
IGA	76.12	61.08	70.85	70.83	64.33	69.39	69.57	61.49	53.02	62.31	31.15	75.34
LWPSO	78.51	84.02	77.56	78.75	69.03	76.70	0.00	40.58	78.43	65.59	79.28	75.31
MPLCS	63.35	61.90	61.62	76.50	65.47	72.06	76.31	54.43	53.84	78.78	61.36	79.07
PBIL	70.65	60.94	68.06	64.98	65.30	71.54	71.54	54.08	53.02	62.31	53.73	75.34
SGA	65.14	69.21	81.69	70.29	71.54	72.24	71.36	54.43	53.26	54.08	61.63	72.36
SSMA	64.00	68.14	68.42	68.84	63.68	67.92	62.17	54.31	62.17	54.08	61.63	72.36
UCS	44.34	44.44	49.50	44.44	44.44	44.44	54.66	54.54	44.53	62.58	62.58	62.58
XCS	80.96	61.22	59.69	72.24	72.41	72.24	71.89	47.04	54.31	72.06	54.08	75.34
AB	59.13	61.49	60.94	70.83	60.67	68.75	67.98	53.73	61.22	69.06	61.08	89.10
BG	65.30	62.04	59.27	67.74	65.63	66.26	72.41	62.44	53.37	80.62	60.94	77.46
C4.5	71.54	62.31	59.83	66.80	66.89	59.97	67.05	53.96	60.94	68.75	61.08	79.92
PART	80.96	53.61	80.13	60.46	73.76	68.11	66.86	31.49	59.41	23.99	61.22	0.00
RIPPER	75.98	62.44	62.85	44.53	62.71	75.34	62.04	71.01	70.83	73.16	70.11	83.41
SLIPPER	72.06	61.90	67.05	69.93	72.06	59.83	66.10	62.44	61.77	78.96	61.63	78.86
CHI-RW	78.32	56.39	75.41	75.73	72.82	70.94	73.08	47.04	47.04	46.54	47.04	76.57
KNN	51.57	61.77	54.54	83.14	59.83	59.27	71.71	54.54	53.02	79.71	61.08	71.71
KSTAR	65.94	62.31	50.46	71.13	69.36	68.60	68.45	53.96	52.90	59.69	54.43	69.75
LRR	85.21	68.91	85.87	77.65	79.71	81.17	79.49	69.51	74.98	0.00	69.51	85.87
PUBLIC	75.54	61.08	53.26	65.85	65.14	71.54	72.06	44.53	58.98	0.00	62.44	85.01
BNGE	52.66	61.63	53.96	62.68	61.08	68.60	74.13	53.96	53.37	73.96	61.08	79.49
EACH	74.75	74.75	74.75	74.75	74.75	74.75	80.16	74.75	74.75	74.75	74.75	74.75
RISE	60.39	94.00	61.63	72.41	60.39	61.08	70.71	53.84	53.61	78.40	68.14	80.34

6.3. RQ3: Which data resampling method best improves the performance of the class maintainability prediction models?

As discussed in Section 6.2, the performance of the class maintainability prediction models has shown a vast positive improvement in terms of GM and BL performance metrics. To carry our investigation further, we are interested in detecting which of the data resampling technique best improves the performance of class maintainability prediction models. To investigate this, we carried out statistical analysis using the Friedman test. We applied the Friedman test by evaluating the GM and BL of prediction models developed after all data resampling techniques on all datasets under examination in this work. The Friedman test is applied at a level of significance α = 0.05 with 12 degrees of freedom (12 data resampling techniques plus when no resampling was used). We tested the following hypothesis using the Friedman test.

Table 14
GM Results of Class Maintainability Prediction Models after Applying Data Resampling Techniques on Ivy Dataset.

Technique	Adasyn	BSMOTE	ROS	Safe-SMOTE	SMOTE	SMOTE-ENN	SMOTE-TL	SD1	SD2	CNN	NCL	RUS
BIOHEL	66.37	53.81	50.86	69.77	58.03	61.22	64.10	55.14	50.16	58.19	60.61	67.84
CHC	73.79	51.63	69.69	68.76	62.97	62.04	65.50	49.49	55.02	32.40	54.68	69.66
CPSO	68.75	69.62	70.36	62.69	68.76	70.50	47.86	45.47	61.68	61.23	45.98	70.10
GGA	63.72	53.80	64.53	67.71	61.21	55.89	50.54	52.03	53.72	37.13	51.47	73.53
GAssist-Int	70.76	49.64	64.28	68.47	61.73	59.94	66.57	81.04	57.90	44.36	57.90	66.93
GAssist-ADI	73.08	53.32	64.41	69.48	62.61	62.98	61.55	52.35	51.79	54.76	54.85	71.07
IGA	66.57	60.00	58.78	65.04	57.94	53.30	58.88	58.62	51.41	37.35	54.42	68.91
LWPSO	71.84	67.69	68.03	60.13	63.68	60.00	45.82	62.24	66.19	49.43	43.29	67.92
MPLCS	65.45	63.66	50.35	67.32	65.05	63.72	52.10	52.35	51.63	41.20	54.16	71.63
PBIL	68.62	54.19	62.90	66.66	65.89	58.52	62.70	55.27	50.57	37.35	51.39	68.91
SGA	66.57	55.71	57.57	62.39	64.09	63.23	59.52	55.02	51.31	36.39	43.65	69.51
SSMA	68.76	53.41	68.87	59.67	64.00	61.80	62.04	58.35	60.35	36.39	43.65	69.51
UCS	26.69	26.69	26.63	26.61	26.69	32.59	70.30	32.59	32.59	37.58	37.58	37.58
XCS	71.68	53.99	62.75	69.77	66.37	67.10	71.17	41.82	45.68	48.07	52.67	72.59
AB	67.76	54.50	51.71	72.41	57.00	59.10	58.13	48.60	54.07	37.52	53.99	66.93
BG	67.07	53.99	62.44	74.98	63.08	63.08	71.52	54.68	57.27	45.41	54.76	36.73
C4.5	66.19	56.54	50.82	70.07	64.46	61.51	25.10	51.63	50.82	81.05	54.50	68.02
PART	62.25	55.98	62.44	69.20	60.20	63.22	55.55	18.73	26.25	0.00	0.00	60.92
RIPPER	51.31	36.68	44.24	60.07	63.33	64.36	56.54	66.57	66.72	69.36	60.57	66.19
SLIPPER	74.10	50.90	62.74	70.27	69.15	67.07	64.68	58.26	53.81	23.45	56.08	73.84
CHI-RW	68.42	64.34	60.92	60.30	64.00	61.63	69.54	26.73	26.73	26.73	26.73	60.61
KNN	65.13	56.12	60.63	60.63	61.09	55.67	63.10	63.03	57.18	61.45	65.02	66.93
KSTAR	62.15	47.07	62.15	67.84	62.87	59.88	65.89	37.19	51.14	44.93	75.63	36.73
LRR	67.18	67.32	67.69	68.10	68.75	70.34	70.33	55.78	65.81	58.80	55.61	66.93
PUBLIC	71.95	51.50	40.69	73.99	68.72	60.25	61.86	51.31	32.55	0.00	37.41	36.73
BNGE	64.46	47.76	41.26	63.51	56.72	53.29	56.12	52.11	54.25	59.57	56.72	36.73
EACH	32.68	32.68	32.68	32.68	32.68	32.68	32.68	27.49	32.68	27.18	32.68	36.73
RISE	56.72	50.98	45.27	67.69	52.84	52.67	56.12	48.67	56.91	52.92	60.40	36.73

Table 15
GM Results of Class Maintainability Prediction Models after Applying Data Resampling Techniques on Jcs Dataset.

Tecnhique	Adasyn	BSMOTE	ROS	Safe-SMOTE	SMOTE	SMOTE-ENN	SMOTE-TL	SD1	SD2	CNN	NCL	RUS
BIOHEL	77.09	75.77	78.92	77.26	75.52	78.27	78.94	79.87	81.11	65.48	82.01	75.03
CHC	86.47	82.57	78.64	81.94	83.76	83.76	85.86	79.49	85.83	71.40	86.41	82.21
CPSO	79.21	66.96	80.97	73.56	74.10	78.01	71.93	70.62	73.30	58.75	81.02	73.56
GGA	84.95	75.03	81.54	81.62	84.95	81.03	82.85	79.49	84.65	85.29	43.69	84.35
GAssist-Int	84.22	72.79	85.71	79.54	78.10	83.16	81.33	74.29	81.45	68.34	81.73	80.89
GAssist-ADI	83.65	75.15	75.73	81.62	78.66	83.37	83.16	74.04	83.37	59.50	84.95	77.71
IGA	77.54	80.10	73.58	74.68	74.75	81.64	81.33	76.03	78.88	88.19	79.21	80.32
LWPSO	80.52	57.92	78.77	73.92	74.68	75.20	76.34	72.93	78.63	60.91	82.76	78.94
MPLCS	83.16	76.44	83.88	79.49	75.15	80.32	81.92	79.87	79.76	66.04	34.91	78.59
PBIL	82.55	79.84	82.83	82.21	83.16	83.76	84.64	77.74	85.83	88.19	85.24	80.32
SGA	83.76	77.71	89.09	80.03	82.50	81.62	83.76	79.76	83.65	81.38	81.45	100.00
SSMA	83.39	82.45	76.82	80.32	85.56	82.85	81.64	78.72	83.46	81.38	81.45	100.00
UCS	67.53	67.53	66.35	67.53	67.53	67.53	67.53	67.79	70.71	53.30	53.30	53.30
XCS	85.53	78.39	65.61	85.56	81.00	82.50	82.85	79.58	81.00	34.19	84.79	79.31
AB	80.03	75.28	67.87	78.66	79.06	81.45	77.74	75.41	77.47	70.11	80.03	73.92
BG	79.21	78.80	76.95	78.88	78.94	80.03	76.67	69.07	79.76	65.18	75.93	73.92
C4.5	80.89	70.48	68.09	78.88	80.03	79.21	79.49	76.94	76.40	73.03	70.85	75.26
PART	66.50	33.14	32.44	75.77	37.80	27.07	48.46	0.00	19.13	73.30	0.00	71.09
RIPPER	71.17	73.98	71.63	73.03	72.07	79.46	78.27	78.66	79.21	59.34	84.35	71.09
SLIPPER	73.28	75.03	72.55	78.10	79.06	78.01	77.74	79.76	79.49	68.88	84.95	74.68
CHI-RW	82.71	84.34	83.37	83.69	82.71	82.39	82.39	82.11	85.35	77.69	84.02	80.05
KNN	66.73	70.25	65.61	65.61	70.85	80.89	79.46	74.04	71.09	55.31	84.35	71.09
KSTAR	69.11	66.96	73.92	78.59	72.52	70.11	69.61	77.74	82.21	64.13	78.54	73.92
LRR	82.25	85.24	83.46	83.16	83.46	84.64	84.02	81.92	84.22	43.08	81.38	75.26
PUBLIC	78.01	75.67	72.79	74.68	76.41	74.48	73.56	80.44	76.98	38.49	84.06	81.17
BNGE	76.44	76.95	75.28	76.40	77.74	80.89	80.03	78.54	80.03	69.66	83.94	73.92
EACH	80.71	80.71	80.71	80.71	80.71	80.71	80.71	80.71	80.71	80.71	80.71	71.09
RISE	66.50	66.96	70.71	70.01	66.96	77.26	76.41	74.36	78.66	61.57	82.25	73.92

Null hypothesis-H_{o1}: The class maintainability prediction models built after pre-processing the imbalanced datasets with all the data resampling techniques (Adasyn, BSMOTE, ROS, Safe-SMOTE, SMOTE, SMOTE-ENN, SMOTE-TL, SD1, SD2, CNN, NCL, RUS) do not show significant difference in the GM performance.

Alternate hypothesis-H_{a1}: The class maintainability prediction models built after pre-processing the imbalanced datasets with all the data resampling techniques (Adasyn, BSMOTE, ROS, Safe-SMOTE, SMOTE, SMOTE-ENN, SMOTE-TL, SD1, SD2, CNN, NCL, RUS) show significant difference in the performance evaluated in terms of performance metric GM.

Null hypothesis-H_{o2}: The class maintainability prediction models built after pre-processing the imbalanced datasets with all the data resampling techniques (Adasyn, BSMOTE, ROS, Safe-SMOTE, SMOTE, SMOTE-ENN, SMOTE-TL, SD1, SD2, CNN, NCL, RUS) do not show significant difference in the performance evaluated in terms of performance metric BL.

Table 16
GM Results of Class Maintainability Prediction Models after Applying Data Resampling Techniques on Lang Dataset.

Technique	Adasyn	BSMOTE	ROS	Safe-SMOTE	SMOTE	SMOTE-ENN	SMOTE-TL	SD1	SD2	CNN	NCL	RUS
BIOHEL	77.09	79.29	74.76	79.03	58.94	76.84	79.42	75.13	80.78	68.29	81.25	76.63
CHC	79.09	83.58	81.01	83.50	82.89	80.79	81.72	81.46	83.35	79.51	82.89	79.82
CPSO	76.05	82.28	82.18	79.58	79.82	83.21	74.65	79.58	78.78	67.75	81.91	81.15
GGA	77.38	81.68	79.38	80.06	62.22	80.79	79.88	81.46	83.58	79.72	81.24	82.53
GAssist-Int	80.30	81.24	79.65	82.89	78.41	84.93	82.42	78.85	82.89	70.91	78.50	82.77
GAssist-ADI	81.79	79.29	82.54	82.65	80.79	82.19	84.70	80.57	82.42	73.17	82.19	83.01
IGA	72.70	77.30	72.45	75.77	78.27	78.12	68.92	73.51	70.19	77.48	70.77	78.85
LWPSO	74.42	76.88	78.77	77.49	79.80	80.01	76.18	75.75	74.65	66.44	72.17	73.98
MPLCS	78.36	81.24	77.30	83.74	80.33	83.12	80.33	83.12	83.12	68.30	79.42	84.97
PBIL	79.29	83.58	71.61	80.11	69.40	81.24	81.72	83.12	79.07	77.48	82.65	78.85
SGA	77.87	81.24	79.38	81.25	79.07	82.65	77.08	81.68	82.89	77.27	49.85	78.61
SSMA	79.03	81.46	82.42	83.74	77.53	82.42	81.48	84.93	83.26	77.27	82.89	78.61
UCS	65.27	65.27	69.42	65.27	62.66	65.27	62.66	65.27	65.45	67.94	67.94	67.94
XCS	79.82	81.46	79.59	82.42	80.33	80.79	82.19	78.63	82.65	65.56	79.88	80.63
AB	72.70	74.75	63.48	82.53	78.63	75.99	77.08	75.56	75.13	75.42	77.75	67.94
BG	75.04	79.07	75.13	83.26	81.01	82.89	80.78	66.61	79.42	68.99	79.88	75.04
C4.5	76.62	83.35	72.03	83.26	84.46	79.88	73.82	79.42	78.50	71.34	79.65	75.04
PART	27.49	83.35	53.44	65.62	80.79	42.47	48.37	32.81	27.00	57.03	0.00	44.76
RIPPER	69.73	59.42	67.75	61.98	65.09	79.29	76.20	77.56	79.09	89.22	78.61	78.19
SLIPPER	75.49	76.84	73.30	79.88	80.11	76.20	73.81	77.75	77.08	66.26	76.18	79.54
CHI-RW	77.90	77.58	81.17	76.80	82.31	81.40	75.78	51.75	51.75	69.16	51.75	85.09
KNN	72.47	76.63	72.78	72.78	75.99	77.08	76.18	74.55	67.98	64.11	79.19	67.94
KSTAR	66.38	69.54	65.97	80.54	71.58	77.05	80.56	76.63	71.45	68.79	79.07	75.04
LRR	85.85	86.69	85.46	86.37	85.21	84.22	81.93	81.46	80.55	81.48	82.89	67.94
PUBLIC	78.12	83.58	76.18	85.64	82.65	82.89	82.19	73.93	72.49	30.19	79.07	75.95
BNGE	65.12	72.18	64.20	77.09	71.38	76.20	78.27	32.85	70.69	64.62	76.63	67.94
EACH	27.15	27.15	27.15	27.15	27.15	27.15	27.15	27.15	23.62	27.15	27.15	75.04
RISE	62.20	64.56	57.95	76.13	74.14	78.63	81.01	70.36	69.73	61.17	76.40	67.94

Table 17
GM Results of Class Maintainability Prediction Models after Applying Data Resampling Techniques on Log4j Dataset.

Technique	Adasyn	BSMOTE	ROS	Safe-SMOTE	SMOTE	SMOTE-ENN	SMOTE-TL	SD1	SD2	CNN	NCL	RUS
BIOHEL	63.36	57.32	63.86	67.33	66.49	65.87	64.80	67.74	70.25	63.90	79.77	71.42
CHC	74.23	76.70	79.77	75.86	77.88	75.28	75.74	62.61	67.53	60.08	72.10	73.69
CPSO	62.99	74.01	55.77	73.56	68.84	71.17	66.43	70.14	62.87	65.35	67.27	75.42
GGA	69.23	71.87	67.08	74.36	70.45	72.22	77.01	62.42	71.53	47.29	74.45	36.68
GAssist-Int	70.58	71.98	71.28	74.91	72.47	73.13	77.07	62.98	62.85	91.05	72.24	70.86
GAssist-ADI	68.16	67.61	69.09	73.85	72.16	75.54	74.76	65.80	63.77	62.63	70.97	69.65
IGA	68.89	73.23	70.73	69.51	71.42	67.88	69.02	66.82	69.29	55.63	63.25	76.45
LWPSO	68.39	71.45	66.10	74.36	69.40	73.76	67.49	60.53	68.97	67.33	67.76	67.71
MPLCS	73.37	71.87	66.92	71.29	70.95	71.05	69.96	60.75	67.74	56.85	67.21	67.08
PBIL	68.73	72.65	73.47	75.32	74.23	71.54	75.56	65.70	68.06	55.63	68.77	76.45
SGA	65.67	67.94	72.34	73.11	70.29	72.22	75.00	66.39	72.66	55.88	68.94	74.30
SSMA	69.91	70.37	69.18	74.23	71.17	77.05	74.09	69.42	65.66	55.88	68.94	74.30
UCS	54.79	54.79	54.43	54.08	54.79	58.52	54.79	54.94	54.94	56.61	56.61	56.61
XCS	72.91	65.70	65.85	73.36	69.63	68.61	72.60	57.13	66.91	62.85	69.74	71.01
AB	64.80	61.12	60.20	64.20	61.47	61.47	64.60	63.35	65.45	61.93	67.94	56.61
BG	69.47	64.32	64.28	69.68	63.15	67.72	72.03	56.89	59.36	61.39	69.01	56.61
C4.5	70.19	60.66	63.67	71.40	64.88	61.65	65.69	63.84	66.99	58.09	61.47	67.95
PART	59.29	55.17	66.41	60.17	63.64	59.95	53.97	21.73	37.48	56.37	21.79	50.53
RIPPER	64.82	57.92	60.57	71.30	59.92	70.87	63.67	65.69	65.34	52.17	71.93	62.31
SLIPPER	63.64	63.93	64.69	67.33	64.92	67.10	67.61	55.40	65.50	63.42	63.98	63.96
CHI-RW	70.94	77.80	76.32	74.51	75.78	76.80	72.58	30.76	57.23	29.71	51.89	70.89
KNN	69.15	66.48	44.00	44.00	70.03	74.23	73.13	17.02	81.57	49.30	60.71	62.31
KSTAR	48.72	61.93	72.28	73.98	54.16	54.39	63.74	56.61	61.21	69.26	54.55	70.89
LRR	72.66	74.23	74.00	72.48	74.29	75.74	74.52	59.47	72.65	47.50	73.00	62.31
PUBLIC	67.10	70.95	71.05	67.19	66.41	69.10	73.27	55.13	54.88	0.00	61.57	86.57
BNGE	61.47	62.23	64.42	67.61	62.36	66.23	64.35	61.21	65.50	63.56	69.66	70.89
EACH	48.57	48.57	44.30	48.57	48.57	48.57	48.57	48.57	48.57	48.57	48.57	74.30
RISE	56.95	60.57	57.68	67.27	60.50	65.13	64.35	68.38	64.71	60.43	69.96	74.30

Alternate hypothesis-H_{a2}: The class maintainability prediction models built after pre-processing the imbalanced datasets with all the data resampling techniques (Adasyn, BSMOTE, ROS, Safe-SMOTE, SMOTE, SMOTE-ENN, SMOTE-TL, SD1, SD2, CNN, NCL, RUS) show significant difference in the performance evaluated in terms of performance metric BL.

Table 27 shows the results of the Friedman test carried out for GM and BL performance metrics. The p-value obtained was 0.000 (i.e., p-value less than 0.05) both for GM and BL, i.e., the results of the Friedman test are significant.

Therefore, we reject the null hypotheses H_{o1} and H_{o2}, which states the performance of class maintainability prediction models after applying different data resampling techniques is the same. The mean ranks assigned to each data resampling technique for performance metrics GM and BL are presented in Table 27. It is to be noted that Safe-SMOTE technique has obtained the best rank

Table 18
GM Results of Class Maintainability Prediction Models after Applying Data Resampling Techniques on Ode Dataset.

Technique	Adasyn	BSMOTE	ROS	Safe-SMOTE	SMOTE	SMOTE-ENN	SMOTE-TL	SD1	SD2	CNN	NCL	RUS
BIOHEL	63.78	54.62	58.32	65.68	62.40	67.39	65.35	52.28	61.69	38.57	55.33	64.91
CHC	69.21	58.03	71.17	71.05	64.25	68.60	69.77	55.65	61.74	48.93	61.99	71.18
CPSO	51.59	62.78	62.39	58.03	55.10	56.15	73.16	57.91	63.23	62.42	55.91	57.26
GGA	59.70	59.38	63.27	68.11	66.07	65.52	65.26	60.96	55.90	50.71	57.61	71.75
GAssist-Int	70.24	57.95	67.82	71.10	70.19	69.89	66.18	43.48	58.58	46.71	49.92	72.90
GAssist-ADI	69.58	64.25	66.08	71.53	71.15	72.55	69.70	47.12	60.87	44.37	51.66	73.94
IGA	57.13	58.55	55.50	66.08	65.92	62.47	66.51	53.28	58.28	49.08	59.53	70.57
LWPSO	56.29	57.22	57.27	61.30	56.89	60.07	52.59	64.71	55.52	54.64	61.42	54.62
MPLCS	68.28	53.74	65.73	69.05	71.00	67.20	70.51	50.24	56.61	46.74	56.33	70.96
PBIL	60.84	63.13	62.13	66.94	62.00	66.94	68.22	58.45	58.47	49.08	59.84	70.57
SGA	58.22	55.60	59.78	68.21	64.12	62.63	69.00	59.85	54.31	48.99	54.45	71.49
SSMA	60.74	60.27	56.47	69.26	66.33	64.02	65.26	58.27	59.30	48.99	54.45	71.49
UCS	41.43	39.45	43.57	41.43	41.48	34.79	41.48	45.38	43.61	37.19	37.19	37.19
XCS	67.29	52.80	58.89	69.90	66.70	66.99	69.81	43.61	46.48	58.87	44.99	67.09
AB	58.97	52.98	51.07	67.50	61.91	66.47	66.89	59.26	55.90	52.67	60.55	71.49
BG	62.30	53.18	55.11	69.41	63.71	68.64	67.68	54.76	54.27	48.31	57.21	67.09
C4.5	65.02	51.30	42.18	69.46	65.94	70.84	67.15	57.50	59.38	44.65	57.83	67.82
PART	45.44	35.62	56.69	56.58	54.41	30.09	46.09	18.71	22.68	0.00	0.00	53.44
RIPPER	57.56	78.61	55.71	72.83	67.77	71.88	67.14	64.79	63.53	52.62	70.12	63.45
SLIPPER	62.13	55.65	53.64	64.01	63.97	65.97	68.63	57.50	52.88	54.70	59.34	67.06
CHI-RW	68.14	85.56	75.52	70.49	71.78	70.06	69.15	30.51	30.51	30.42	37.82	69.98
KNN	55.79	61.89	50.05	49.16	62.59	62.00	62.77	50.02	53.96	44.65	54.41	67.06
KSTAR	50.84	50.74	62.17	65.19	60.59	62.25	65.35	0.00	18.66	32.13	0.00	63.45
LRR	71.32	68.92	72.66	71.43	73.00	72.78	73.10	47.03	57.24	41.05	47.00	67.06
PUBLIC	72.31	54.10	51.33	66.73	68.89	72.21	67.21	58.75	60.96	0.00	52.74	65.66
BNGE	57.56	69.06	61.15	64.17	72.46	71.49	69.16	50.40	56.72	46.00	57.06	67.09
EACH	62.25	32.39	32.39	32.39	32.39	32.39	32.39	32.39	32.39	32.39	32.39	63.45
RISE	38.74	63.11	54.31	64.08	65.61	66.94	66.99	48.60	51.63	53.11	60.07	63.45

Table 19
BL Results of Class Maintainability Prediction Models after Applying Data Resampling Techniques on Bcel Dataset.

Technique	Adasyn	BSMOTE	ROS	Safe-SMOTE	SMOTE	SMOTE-ENN	SMOTE-TL	SD1	SD2	CNN	NCL	RUS
BIOHEL	73.56	72.14	71.69	75.67	69.78	64.61	67.44	64.46	64.42	69.29	72.17	75.56
CHC	70.80	72.06	69.97	78.41	71.41	64.61	73.20	64.53	68.39	68.22	72.24	73.81
CPSO	63.37	65.09	63.76	64.65	63.56	68.33	59.72	64.00	64.30	47.18	63.53	61.74
GGA	70.54	72.24	72.44	78.6	74.87	64.61	69.59	64.49	72.21	68.08	72.21	73.61
GAssist-Int	75.43	72.06	74.92	78.51	75.43	64.61	75.23	64.56	72.17	64.46	72.17	74.60
GAssist-ADI	76.05	72.02	71.77	76.05	76.05	64.62	72.17	64.61	62.19	42.49	60.68	78.41
IGA	73.20	71.89	74.91	71.64	70.79	60.69	72.51	56.60	63.79	63.74	44.93	73.81
LWPSO	63.40	65.09	65.58	64.2	64.00	65.76	60.39	65.96	74.31	54.10	61.51	55.25
MPLCS	76.05	72.17	74.13	76.05	76.05	64.62	76.05	60.66	64.44	60.61	60.68	78.68
PBIL	71.70	72.02	71.37	78.41	71.64	64.61	73.20	64.53	72.17	63.74	72.21	73.81
SGA	70.67	72.10	69.95	78.41	70.72	64.61	73.51	68.34	68.37	68.39	72.30	76.21
SSMA	65.35	71.80	65.55	80.84	75.37	56.76	47.72	68.51	72.06	68.39	72.30	76.21
UCS	41.07	41.07	41.07	41.07	41.07	41.07	41.07	41.07	41.07	37.15	37.15	37.15
XCS	67.09	69.65	61.75	60.5	53.50	64.61	53.46	52.85	56.77	48.9	64.56	70.56
AB	73.57	72.10	57.95	75.3	71.69	68.51	80.76	60.64	64.49	54.82	72.24	77.17
BG	73.27	72.02	68.46	75.02	74.37	68.51	73.51	60.62	64.33	67.82	72.27	75.82
C4.5	72.81	72.02	68.24	75.23	71.75	60.66	70.79	60.64	68.12	52.44	68.39	78.23
PART	72.02	72.02	71.80	56.74	60.51	29.29	74.00	29.29	29.29	29.29	29.29	29.29
RIPPER	0.00	37.15	0.00	33.21	0.00	68.51	48.92	52.18	53.51	50.39	53.90	79.31
SLIPPER	70.14	68.22	72.02	72.1	72.02	68.51	75.09	68.37	68.46	64.01	76.16	76.21
CHI-RW	72.35	52.85	56.77	56.76	56.76	45.00	72.27	37.14	37.14	29.29	37.14	56.75
KNN	56.77	52.85	41.06	48.92	52.85	56.78	76.25	91.09	60.59	63.50	72.24	41.15
KSTAR	70.94	68.22	54.11	74.46	69.50	64.61	72.42	64.49	64.44	47.20	60.64	75.02
LRR	77.28	68.44	75.61	78.85	78.94	64.62	76.46	64.59	68.48	37.14	64.59	75.82
PUBLIC	75.16	72.02	61.33	78.23	69.30	64.62	69.30	68.34	68.29	55.58	64.56	71.58
BNGE	68.85	64.44	61.94	67.98	65.92	68.52	65.06	59.85	62.85	38.91	64.3	75.29
EACH	57.86	57.86	57.86	57.86	57.86	57.86	57.86	57.86	57.86	57.86	57.86	57.86
RISE	66.82	68.22	54.26	78.23	69.20	64.61	83.67	68.39	68.34	52.60	68.34	69.09

(lowest rank values) for GM and BL performance metrics. The data resampling techniques RUS, SMOTE-ENN, SMOTE-TL, and SMOTE are amongst top five rankers as per Friedman test analysis for GM and BL. The performance of the class maintainability models is very poor when datasets are imbalanced as the no-resampling scenario has obtained the worst rank in Friedman test analysis (Table 12).

As per the results of the Friedman test, Safe-SMOTE was the best data resampling technique that improved the performance of class maintainability prediction models for GM and BL. To further examine and confirm, whether Safe-SMOTE is superior to other data resampling techniques used in this work, we conducted post-hoc analysis by Wilcoxon-signed rank test with Bonferroni correction to take care of the family-wise error. The performance

Table 20
BL Results of Class Maintainability Prediction Models after Applying Data Resampling Techniques on Betwixt Dataset.

Technique	Adasyn	BSMOTE	ROS	Safe-SMOTE	SMOTE	SMOTE-ENN	SMOTE-TL	SD1	SD2	CNN	NCL	RUS
BIOHEL	49.71	49.73	48.85	62.51	52.04	56.3	58.94	46.47	46.73	47.39	53.22	64.91
CHC	72.27	53.89	69.85	69.02	69.28	70.22	65.53	67.10	67.85	29.28	57.73	71.17
CPSO	65.42	66.94	59.46	65.89	63.91	70.06	58.32	60.29	60.77	61.62	54.07	54.28
GGA	63.72	41.72	56.88	67.89	65.97	62.21	64.85	64.76	58.73	37.11	53.94	71.73
GAssist-Int	64.49	41.82	60.35	67.43	64.48	64.69	64.70	49.42	59.53	49.58	47.76	72.89
GAssist-ADI	60.29	43.97	65.08	64.22	65.05	63.12	59.90	49.36	52.05	49.86	49.02	73.92
IGA	57.98	56.23	62.16	63.59	59.24	57.19	61.26	50.75	47.95	31.89	56.20	70.55
LWPSO	65.42	65.35	61.31	63.73	67.05	58.62	51.69	65.62	59.78	68.13	59.64	51.41
MPLCS	62.70	41.67	46.16	65.66	61.71	57.71	58.71	52.17	51.73	41.98	52.74	70.95
PBIL	61.97	52.57	66.61	69.39	64.49	67.4	66.35	59.45	56.59	31.89	56.28	70.55
SGA	64.68	47.00	55.88	67.65	67.89	62.44	65.53	54.36	59.37	42.29	51.43	71.47
SSMA	64.49	49.73	56.26	67.4	68.38	68.76	66.40	56.98	56.50	42.29	51.43	71.47
UCS	37.13	37.13	34.52	37.13	37.13	37.13	37.13	37.13	34.52	37.06	39.21	39.21
XCS	57.45	41.82	45.52	65.72	66.09	68.63	53.12	47.09	46.67	55.26	44.11	66.97
AB	57.12	51.99	43.64	59.22	47.46	56.72	63.37	49.10	46.34	34.31	57.33	71.47
BG	66.62	39.58	56.32	68.63	61.10	58.53	59.86	44.03	45.63	44.09	53.85	66.97
C4.5	60.12	44.50	47.96	45.48	61.23	54.79	64.69	41.34	45.17	37.08	53.98	67.81
PART	56.86	48.66	29.29	59.64	34.51	53.38	34.27	29.29	29.29	34.46	29.29	52.48
RIPPER	45.96	47.37	43.71	63.52	53.15	54.89	55.73	60.29	58.18	58.06	68.23	62.02
SLIPPER	56.99	52.27	36.32	67.75	61.41	57.87	63.86	49.10	46.26	51.87	56.14	67.00
CHI-RW	70.89	65.77	68.35	59.70	69.29	71.08	69.27	32.07	48.63	29.26	39.56	69.53
KNN	57.62	64.94	44.31	44.03	67.18	60.46	62.29	49.36	53.8	53.93	52.79	67.00
KSTAR	56.01	43.84	45.03	47.97	46.08	48.23	48.95	36.67	39.87	33.90	29.29	62.02
LRR	71.32	36.00	70.28	71.82	72.06	72.06	68.23	54.28	62.83	42.21	45.37	67.00
PUBLIC	68.40	49.68	58.53	59.66	68.14	60.77	62.91	56.86	61.11	29.29	50.15	64.92
BNGE	62.29	61.32	56.75	51.51	65.05	59.08	61.44	49.42	52.22	58.63	53.82	66.97
EACH	37.11	37.11	37.11	37.11	37.11	37.11	37.11	37.11	37.11	37.11	36.73	62.02
RISE	53.80	49.23	39.26	63.81	60.78	60.12	60.85	46.78	51.73	59.95	57.16	62.02

Table 21
BL Results of Class Maintainability Prediction Models after Applying Data Resampling Techniques on Io Dataset.

Technique	Adasyn	BSMOTE	ROS	Safe-SMOTE	SMOTE	SMOTE-ENN	SMOTE-TL	SD1	SD2	CNN	NCL	RUS
BIOHEL	56.88	50.43	50.01	66.88	50.18	57.31	70.76	77.42	56.25	70.52	57.27	80.33
CHC	69.20	64.40	66.79	69.55	69.65	69.43	75.54	50.49	50.35	43.42	57.42	73.40
CPSO	73.86	75.81	77.145	79.70	73.04	80.13	57.22	77.89	83.55	69.02	88.39	79.20
GGA	69.43	57.47	60.025	62.98	70.16	63.87	63.15	50.47	57.31	50.37	50.48	71.11
GAssist-Int	62.61	57.45	53.135	69.55	70.16	63.31	69.86	57.55	50.48	64.33	57.47	78.16
GAssist-ADI	62.10	64.47	66.825	75.97	69.09	62.79	69.65	50.49	50.46	63.81	57.24	74.10
IGA	75.54	57.31	67.6	69.43	62.61	68.46	68.59	57.40	50.3	57.52	36.33	74.93
LWPSO	76.62	79.20	76.62	76.91	69.02	76.54	58.69	42.07	78.39	64.39	79.27	72.92
MPLCS	61.99	57.47	59.605	75.83	63.23	70.16	75.68	50.49	50.45	77.32	57.37	79.06
PBIL	69.32	57.27	64.25	62.98	63.15	69.86	70.06	50.47	50.3	57.52	50.43	74.93
SGA	63.06	64.52	79.105	69.09	69.86	70.25	69.76	50.49	50.35	50.47	57.42	72.29
SSMA	62.41	64.29	64.43	68.05	62.20	67.33	61.17	50.49	57.51	50.47	57.42	72.29
UCS	43.42	43.42	46.96	43.42	43.42	43.42	50.49	50.50	43.43	57.55	57.55	57.55
XCS	80.94	57.34	56.875	70.25	70.34	70.25	70.06	45.00	50.49	70.16	50.47	74.93
AB	56.65	57.4	57.27	69.43	57.20	64.43	64.24	50.43	57.34	64.49	57.31	86.01
BG	63.15	57.49	56.71	67.19	63.31	63.61	70.34	57.54	50.37	78.19	57.27	76.50
C4.5	69.86	57.52	56.93	66.41	63.87	56.98	63.93	50.46	57.27	64.43	57.31	79.92
PART	80.94	50.41	80.13	59.86	73.58	67.48	66.79	36.36	56.77	29.64	57.34	29.29
RIPPER	71.59	57.54	57.56	43.43	57.56	74.93	57.49	69.55	69.43	73.04	68.96	83.12
SLIPPER	70.16	57.47	63.93	68.84	70.16	56.93	63.54	57.54	57.45	77.42	57.42	78.84
CHI-RW	72.67	52.75	75.33	75.24	70.42	70.51	70.7	45.00	45.00	44.97	45.00	74.68
KNN	49.86	57.45	50.50	78.77	56.93	56.71	69.97	50.50	50.30	77.80	57.31	71.06
KSTAR	63.46	57.52	49.36	71.11	64.54	64.40	64.36	50.46	50.27	56.88	50.49	68.72
LRR	84.61	64.47	85.40	76.63	79.70	81.14	79.49	64.56	71.37	29.20	64.56	85.4
PUBLIC	75.08	57.31	50.35	65.58	63.06	69.86	70.16	43.43	56.59	29.29	57.54	80.39
BNGE	50.21	57.42	50.46	61.53	57.31	64.40	71.10	50.46	50.37	71.04	57.31	79.49
EACH	74.44	74.44	74.44	74.44	74.44	74.44	78.01	74.44	74.44	74.44	74.44	74.44
RISE	57.12	93.49	57.42	70.34	57.12	57.31	64.64	50.45	50.41	77.10	64.29	80.33

of models employed after Safe-SMOTE and other data-resampling techniques is compared in terms of GM and BL.

In this work, for the Wilcoxon-signed rank test, the null and alternative hypothesis are given as follows:

H_{o3}: $GM_{\text{safe-SMOTE}} = GM_X$
H_{a3}: $GM_{\text{safe-SMOTE}} \neq GM_X$

where X denotes Adasyn, BSMOTE, ROS, SMOTE-ENN, SMOTE-TL, SMOTE, SD1, SD2, CNN, NCL, RUS. Similarly, the null and alternate hypothesis for BL are given as follows:

H_{o4}: $BL_{\text{safe-SMOTE}} = BL_X$
H_{a4}: $BL_{\text{safe-SMOTE}} \neq BL_X$

Table 22
BL Results of Class Maintainability Prediction Models after Applying Data Resampling Techniques on Ivy Dataset.

Technique	Adasyn	BSMOTE	ROS	Safe-SMOTE	SMOTE	SMOTE-ENN	SMOTE-TL	SD1	SD2	CNN	NCL	RUS
BIOHEL	64.84	51.51	49.08	68.88	55.90	58.63	62.46	53.34	48.79	47.39	53.22	67.82
CHC	73.77	51.13	69.49	68.75	62.94	61.90	65.27	47.19	51.84	29.28	57.73	69.63
CPSO	68.69	69.59	70.11	62.66	68.75	69.93	46.89	44.39	60.81	61.62	54.07	70.09
GGA	62.98	52.61	63.68	67.27	60.45	55.28	50.45	49.36	51.48	37.11	53.94	73.24
GAssist-Int	70.59	49.26	63.72	68.47	61.34	59.74	66.56	77.16	54.34	49.58	47.76	66.87
GAssist-ADI	72.23	52.31	63.51	69.45	62.09	62.39	61.54	49.41	49.31	49.86	49.02	70.92
IGA	65.79	58.93	57.36	64.91	57.36	52.81	58.77	56.16	50.40	31.89	56.20	68.85
LWPSO	71.73	67.58	68.00	58.45	62.57	56.41	45.66	59.03	65.50	68.13	59.64	67.55
MPLCS	65.28	62.19	49.77	66.93	64.57	62.98	51.83	49.41	49.27	41.98	52.74	71.17
PBIL	68.01	52.84	61.27	66.36	64.54	58.04	62.52	51.89	48.95	31.89	56.28	68.85
SGA	65.79	54.58	55.775	61.33	63.26	63.00	59.07	51.84	49.19	42.29	51.43	69.47
SSMA	68.58	52.37	66.235	58.68	62.41	60.9	61.90	54.42	56.76	42.29	51.43	69.47
UCS	34.34	34.34	34.335	34.34	34.34	36.86	64.64	36.86	36.86	37.06	39.21	39.39
XCS	71.57	52.72	60.63	68.88	64.84	65.64	71.05	41.90	44.41	55.26	44.11	72.53
AB	65.65	51.72	49.29	72.16	54.10	56.36	55.94	46.82	51.59	34.31	57.33	66.87
BG	65.27	51.57	59.09	74.81	60.80	60.8	70.07	51.76	54.18	44.09	53.85	39.57
C4.5	65.50	53.94	49.04	69.72	62.68	59.33	33.49	49.27	49.04	37.08	53.98	67.22
PART	62.02	55.88	62.42	68.92	59.98	63.10	52.52	31.80	34.29	34.46	29.29	60.87
RIPPER	49.19	39.25	44.80	56.68	59.34	64.27	53.94	63.74	65.06	58.06	68.23	65.50
SLIPPER	72.72	49.06	59.18	69.25	67.57	65.27	62.81	54.41	51.51	51.87	56.14	73.58
CHI-RW	67.11	62.61	58.50	58.20	62.41	58.80	69.26	34.34	34.34	29.26	39.56	58.35
KNN	63.06	54.84	56.83	56.83	60.36	55.10	62.88	59.27	54.15	53.93	52.79	66.87
KSTAR	61.16	46.36	61.16	67.37	60.69	57.98	64.54	39.35	49.14	33.9	29.29	39.57
LRR	66.82	66.35	66.61	67.59	68.11	69.95	70.27	51.96	61.82	42.21	45.37	66.87
PUBLIC	71.52	50.45	41.69	73.75	68.54	59.68	61.45	49.19	37.18	29.29	50.15	39.57
BNGE	61.40	46.61	41.82	61.00	54.01	51.31	54.84	49.37	51.65	58.63	53.82	39.57
EACH	36.87	36.87	36.87	36.87	36.87	36.87	36.87	33.52	36.87	37.11	36.73	39.57
RISE	54.01	49.09	44.36	66.61	51.13	51.05	54.84	46.84	54.07	59.95	57.16	39.57

Table 23
BL Results of Class Maintainability Prediction Models after Applying Data Resampling Techniques on Jcs Dataset.

Technique	Adasyn	BSMOTE	ROS	Safe-SMOTE	SMOTE	SMOTE-ENN	SMOTE-TL	SD1	SD2	CNN	NCL	RUS
BIOHEL	76.34	70.69	68.205	77.10	72.98	77.14	78.50	72.87	80.04	65.19	81.60	74.72
CHC	86.29	82.02	85.53	81.91	83.71	83.71	85.60	84.02	85.81	68.04	86.36	82.20
CPSO	78.83	64.64	71.44	72.93	74.10	78.01	68.83	70.62	72.72	56.72	80.69	72.93
GGA	84.54	75.00	87.07	81.62	84.54	81.03	82.73	79.32	84.64	83.63	44.10	84.34
GAssist-Int	84.02	70.13	82.91	79.47	77.82	83.06	81.33	74.56	81.16	67.53	81.38	80.69
GAssist-ADI	83.53	75.27	82.21	81.62	78.28	83.28	83.06	73.71	83.28	58.83	84.94	77.71
IGA	76.71	78.14	68.98	74.68	74.49	81.38	81.04	74.24	78.86	84.29	78.72	80.19
LWPSO	79.32	59.36	75.72	73.76	74.68	74.72	72.01	72.93	78.5	59.91	81.98	78.83
MPLCS	83.06	72.52	80.42	78.92	74.07	80.19	81.91	79.87	79.13	64.15	81.16	78.58
PBIL	82.40	78.02	85.60	82.06	83.06	83.71	84.18	81.60	85.81	84.29	85.24	80.19
SGA	83.71	77.61	85.92	79.93	82.48	81.62	83.52	81.60	83.53	78.64	81.16	65.24
SSMA	82.72	74.86	85.00	80.19	85.25	82.73	81.38	78.72	83.39	78.64	81.16	65.24
UCS	63.14	63.14	65.62	63.14	63.14	63.14	63.14	63.14	67.79	50.16	50.16	50.16
XCS	85.53	76.85	62.49	85.25	80.65	82.48	82.73	79.13	80.65	37.03	84.49	79.20
AB	79.32	72.87	65.05	78.28	77.61	81.16	76.79	74.24	76.61	68.77	79.32	73.76
BG	78.72	77.46	75.14	78.86	78.50	79.93	76.03	67.03	79.13	62.40	74.56	73.76
C4.5	80.69	67.69	65.14	78.86	79.32	78.72	78.92	76.23	75.82	72.51	69.23	75.24
PART	64.40	37.14	37.03	73.08	39.71	34.64	48.00	29.29	31.90	72.72	29.29	69.37
RIPPER	67.96	70.69	68.11	71.67	69.90	79.41	77.14	78.28	78.72	59.34	84.34	69.37
SLIPPER	71.82	72.76	70.13	77.82	77.61	76.97	76.79	79.13	78.92	68.84	84.94	74.68
CHI-RW	80.91	82.89	81.70	82.10	80.91	80.51	80.51	81.49	85.09	76.10	82.67	77.70
KNN	64.52	67.59	62.58	62.58	69.23	80.69	79.41	72.26	69.37	55.17	84.34	69.37
KSTAR	68.09	64.64	73.76	78.58	71.34	68.77	68.44	76.79	82.20	63.98	77.30	73.76
LRR	82.06	85.24	83.39	83.06	83.39	84.18	83.46	80.50	84.02	43.78	80.20	75.24
PUBLIC	78.01	74.40	70.23	74.68	76.33	74.25	72.93	80.42	76.85	40.67	84.02	80.08
BNGE	74.86	75.14	72.87	75.82	76.79	80.69	79.93	77.30	79.32	69.19	83.78	73.76
EACH	80.34	80.34	80.34	80.34	80.34	80.34	80.34	80.34	80.34	80.34	80.34	69.37
RISE	64.40	64.64	67.79	69.90	64.64	77.10	76.33	73.52	78.28	61.37	82.06	73.76

The level of significance α = 0.05, with Bonferroni correction was taken for rejecting the null hypotheses Ho3 and Ho4. Because we are comparing 12 pairs of resampling techniques with the Wilcoxon test, so with Bonferroni correction if p-value obtained would be greater than 0.004 (i.e., 0.05/12 = 0.004), the null hypotheses H_{o3} or H_{o4} would be rejected The results of Wilcoxon signed-rank on evaluating the GM and BL performance of class maintainability prediction models after employing Safe-SMOTE and other data resampling techniques are shown in Table 28.

Table 28 shows the test statistics obtained after applying the Wilcoxon signed-rank test on the pair-wise performance of Safe-SMOTE and all other data resampling techniques used in the study. In Table 28, Sig indicates the significant difference in the performance of a pair of compared techniques and NotSig denoted that

Table 24
BL Results of Class Maintainability Prediction Models after Applying Data Resampling Techniques on Lang Dataset.

Techique	Adasyn	BSMOTE	ROS	Safe-SMOTE	SMOTE	SMOTE-ENN	SMOTE-TL	SD1	SD2	CNN	NCL	RUS
BIOHEL	76.34	76.09	72.62	79.01	73.31	73.44	77.80	72.81	79.82	68.26	80.12	76.53
CHC	78.63	81.23	78.51	82.66	80.96	78.43	80.39	78.66	81.15	76.15	80.96	79.17
CPSO	75.66	81.81	82.155	78.99	82.67	83.14	72.48	79.51	78.76	65.75	81.69	81.15
GGA	77.2	78.73	77.025	79.34	80.75	78.43	78.04	78.66	81.23	76.2	78.59	81.99
GAssist-Int	79.51	78.59	77.135	80.96	81.06	83.47	80.75	75.95	80.96	70.81	77.28	82.17
GAssist-ADI	81.43	76.09	80.8	81.43	78.51	80.64	83.35	78.24	80.75	72.63	80.64	82.34
IGA	72.25	75.32	71.895	75.35	77.13	77.84	66.94	71.00	69.62	73.6	67.81	78.43
LWPSO	69.84	76.58	78.61	77.49	80.35	78.71	74.23	74.65	71.67	62.63	70.59	72.64
MPLCS	78.04	78.59	74.51	82.81	80.75	81.06	78.24	81.06	81.06	68.04	77.80	84.62
PBIL	79.25	81.23	70.315	78.14	75.62	78.59	80.39	81.06	76.02	73.6	80.86	78.43
SGA	77.63	78.59	77.025	80.12	76.09	80.86	75.21	78.73	80.96	73.56	80.96	77.94
SSMA	79.01	78.66	80.74	82.81	81.06	80.75	80.26	83.47	82.50	73.56	80.96	77.94
UCS	60.61	60.61	64.56	60.61	60.61	60.61	58.02	60.61	60.63	63.22	63.22	63.22
XCS	79.17	78.66	77.09	80.75	78.24	78.43	80.64	75.88	80.86	65.20	78.04	80.62
AB	71.45	70.91	60.17	81.99	75.88	73.16	75.21	72.99	72.81	75.05	75.52	63.22
BG	73.99	76.02	72.81	82.50	78.51	80.96	79.82	63.16	77.8	68.00	78.04	73.99
C4.5	75.99	81.15	69.88	82.50	83.23	78.04	73.82	77.80	77.28	70.51	77.92	73.99
PART	34.85	81.15	50.17	60.65	78.43	42.35	47.65	37.11	34.52	57.03	29.29	44.56
RIPPER	65.67	55.36	63.19	57.90	60.58	79.25	73.24	76.67	78.63	89.11	78.24	75.71
SLIPPER	74.29	73.44	70.45	78.04	78.14	73.24	72.13	75.52	75.21	65.86	74.71	79.48
CHI-RW	77.77	77.57	79.73	72.65	81.63	79.85	74.81	48.33	48.33	68.57	48.33	84.55
KNN	71.31	73.38	68.4	68.40	73.16	75.21	74.71	70.86	67.25	64.07	77.68	63.22
KSTAR	64.33	65.62	64.11	79.67	68.10	73.50	78.34	73.38	71.19	68.75	76.02	73.99
LRR	85.84	85.80	84.99	86.32	84.81	84.02	81.93	78.66	80.54	80.26	80.96	63.22
PUBLIC	77.84	81.23	74.71	83.78	80.86	80.96	80.64	70.67	72.43	36.04	76.02	74.58
BNGE	62.38	68.27	60.38	76.34	68.03	73.24	77.13	37.33	70.02	64.60	74.97	63.22
EACH	34.53	34.53	34.53	34.53	34.53	34.53	34.53	34.53	32.24	34.53	34.53	73.99
RISE	59.66	60.47	54.99	76.07	70.74	75.88	79.97	67.64	69.26	61.01	74.84	63.22

Table 25
BL Results of Class Maintainability Prediction Models after Applying Data Resampling Techniques on Log4j Dataset.

Technique	Adasyn	BSMOTE	ROS	Safe-SMOTE	SMOTE	SMOTE-ENN	SMOTE-TL	SD1	SD2	CNN	NCL	RUS
BIOHEL	61.33	54.19	59.55	66.36	63.71	63.43	63.52	65.21	68.94	62.90	79.33	71.19
CHC	74.23	76.26	79.73	75.80	77.84	75.07	75.63	59.15	65.11	56.07	70.01	73.63
CPSO	62.92	73.69	55.92	73.50	67.95	70.63	64.19	68.86	61.67	63.17	66.74	75.41
GGA	69.17	69.89	65.61	74.25	69.63	71.85	76.97	59.09	69.70	45.96	72.93	39.39
GAssist-Int	70.48	69.95	70.23	74.42	71.01	72.09	77.06	59.25	60.25	91.05	70.86	70.83
GAssist-ADI	68.15	65.94	67.52	73.15	71.39	74.91	74.60	62.41	61.55	60.92	68.56	69.64
IGA	68.69	72.67	69.43	69.51	70.82	67.22	68.81	64.84	67.66	52.59	61.28	76.38
LWPSO	68.15	71.20	65.75	74.25	67.73	73.47	67.20	59.01	68.92	67.26	67.13	67.60
MPLCS	73.13	69.89	64.82	70.24	69.37	70.07	69.26	57.45	65.21	54.93	64.95	67.08
PBIL	68.54	71.75	72.59	75.29	74.12	70.92	75.55	62.37	65.35	52.59	66.86	76.38
SGA	65.38	66.14	70.91	72.90	69.91	71.85	74.98	62.59	70.29	52.65	66.67	74.25
SSMA	69.59	69.01	67.58	74.23	70.63	76.89	74.00	65.86	63.33	52.65	66.67	74.25
UCS	51.13	51.13	51.07	51.02	51.13	56.23	51.13	51.14	51.14	52.78	52.78	52.78
XCS	72.27	62.37	62.96	72.77	67.86	67.23	72.17	53.33	63.88	60.25	67.92	70.93
AB	63.52	57.56	57.26	64.05	58.73	58.73	62.76	60.46	63.22	57.75	66.14	52.78
BG	68.88	60.81	61.80	69.14	61.22	66.01	71.30	54.06	56.93	57.63	65.72	52.78
C4.5	70.18	57.42	61.50	71.34	64.07	59.66	64.09	60.65	64.84	54.38	58.73	67.45
PART	58.60	53.97	61.20	60.13	63.53	59.10	50.80	32.65	39.38	52.74	32.66	50.04
RIPPER	62.88	54.34	57.39	69.57	57.16	70.78	61.50	64.67	64.42	52.13	71.84	60.73
SLIPPER	62.18	60.68	61.98	66.36	62.94	64.89	65.94	55.24	62.30	62.04	61.65	63.96
CHI-RW	69.19	77.69	75.74	73.69	75.49	76.60	71.01	36.15	54.55	35.83	49.67	69.64
KNN	68.92	65.23	44.02	44.02	69.91	74.12	73.12	31.19	80.60	48.87	59.66	60.73
KSTAR	47.39	57.75	71.48	73.24	51.04	51.08	60.61	52.78	57.58	66.83	51.1	69.64
LRR	71.54	74.12	73.76	72.47	74.07	75.63	74.03	55.93	71.75	46.00	70.44	60.73
PUBLIC	66.22	69.37	70.07	66.87	66.37	68.58	72.96	52.45	52.37	29.29	57.67	84.50
BNGE	58.73	59.02	60.84	65.94	60.02	64.42	63.22	57.58	62.3	61.44	67.52	69.64
EACH	47.34	47.34	44.88	47.34	47.34	47.34	47.34	47.34	47.34	47.34	47.34	74.25
RISE	54.98	57.39	55.32	66.74	58.3	63.06	63.22	65.48	62.82	58.95	69.26	74.25

there is no significant difference in the corresponding pair of techniques. The results of the Wilcoxon signed-rank test indicate that Safe-SMOTE technique is significantly superior to Adasyn, BSMOTE, ROS, SMOTE-ENN, SMOTE-TL, SMOTE, SD1, SD2, CNN, NCL, and No-Resampling scenario in terms of GM and BL performance metrics. However, RUS technique that was amongst the top five rankers according to the Friedman test results in terms of GM and BL, is not found significantly different (p-value > 0.004) in the performance in terms of GM and BL according to the results of Wilcoxon signed-rank test i.e., performance of RUS is comparable with Safe-SMOTE.

6.4. RQ4: What is the comparative performance of the ML and SB techniques after data resampling for developing class maintainability prediction models?

In this section, we compare the different classification techniques after data resampling. To make this comparison, we

Table 26
BL Results of Class Maintainability Prediction Models after Applying Data Resampling Techniques on Ode Dataset

Technique	Adasyn	BSMOTE	ROS	Safe-SMOTE	SMOTE	SMOTE-ENN	SMOTE-TL	SD1	SD2	CNN	NCL	RUS
BIOHEL	62.03	51.46	59.54	64.59	60.72	66.17	64.67	50.69	59.22	53.22	53.22	67.82
CHC	69.03	55.66	70.84	70.84	63.65	68.33	69.72	51.60	57.64	57.73	57.73	69.63
CPSO	50.50	62.77	60.12	57.00	54.01	54.8	71.53	55.60	62.44	54.07	54.07	70.09
GGA	58.87	56.15	67.42	67.42	65.23	65.07	65.23	56.50	52.65	53.94	53.94	73.24
GAssist-Int	70.24	55.62	63.72	70.37	69.66	69.63	66.11	42.92	55.08	47.76	47.76	66.87
GAssist-ADI	68.98	60.96	69.12	70.69	70.02	71.95	69.7	45.38	57.43	49.02	49.02	70.92
IGA	56.09	55.87	62.54	64.86	65.11	61.85	66.49	50.26	55.00	56.20	56.20	68.85
LWPSO	53.44	57.19	61.09	60.03	56.05	59.23	51.02	63.52	54.89	59.64	59.64	67.55
MPLCS	67.20	51.24	66.07	68.13	69.91	66.41	70.49	47.82	53.69	52.74	52.74	71.17
PBIL	59.73	60.52	66.49	66.48	60.87	65.13	68.22	54.06	55.05	56.28	56.28	68.85
SGA	57.18	53.34	65.38	67.50	63.42	61.97	69.00	55.30	51.39	51.43	51.43	69.47
SSMA	59.63	57.24	68.89	68.89	65.42	63.03	65.23	54.04	56.13	51.43	51.43	69.47
UCS	41.67	40.45	40.44	41.67	41.68	37.96	41.68	44.15	42.93	39.21	39.21	39.39
XCS	66.10	50.42	63.32	68.54	64.73	65.89	69.35	42.93	45.29	44.11	44.11	72.53
AB	56.70	50.2	48.89	65.77	59.31	65.13	65.82	55.22	52.65	57.33	57.33	66.87
BG	60.13	50.24	52.44	68.39	61.43	67.44	67.09	51.49	51.39	53.85	53.85	39.57
C4.5	62.70	48.95	42.61	68.43	63.77	70.17	66.66	53.92	56.15	53.98	53.98	67.22
PART	45.47	38.86	56.64	54.39	54.09	35.98	46.14	31.77	32.99	29.29	29.29	60.87
RIPPER	56.83	76.29	53.97	72.59	66.83	70.49	66.00	62.58	61.34	68.23	68.23	65.50
SLIPPER	60.03	52.59	51.21	62.64	62.13	64.34	67.82	53.92	50.18	56.14	56.14	73.58
CHI-RW	68.10	84.69	72.66	69.74	71.31	69.98	68.73	35.95	35.95	39.56	39.56	58.35
KNN	54.01	58.57	47.79	47.17	60.82	60.87	62.50	47.78	51.30	52.79	52.79	66.87
KSTAR	48.83	48.80	60.06	63.37	57.34	59.46	63.95	29.29	31.77	29.29	29.29	39.57
LRR	71.30	67.13	72.44	70.94	72.74	72.70	72.92	45.37	53.86	45.37	45.37	66.87
PUBLIC	72.26	51.34	49.70	65.71	68.33	72.04	67.17	55.94	58.86	50.15	50.15	39.57
BNGE	56.83	65.86	57.50	60.93	71.75	70.99	69.11	47.84	52.80	53.82	53.82	39.57
EACH	62.02	36.73	36.73	36.73	36.73	36.73	36.73	36.73	36.73	36.73	36.73	39.57
RISE	40.28	58.88	51.39	60.90	63.60	65.86	66.52	46.59	49.01	57.16	57.16	39.57

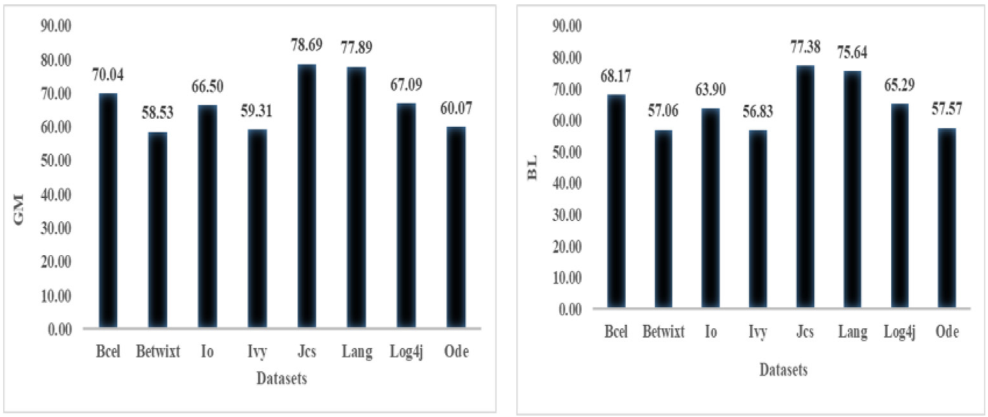

Fig. 3. (a) Median GM values of class maintainability prediction models developed after data resampling. (b) Median BL values of class maintainability prediction models developed after data resampling.

Table 27
Results of Friedman Test.

Data resampling Technique	Mean Rank with respect to GM	Mean Rank with respect to BL
Safe-SMOTE	4.41 (Rank 1)	4.24 (Rank 1)
RUS	4.88 (Rank 2)	5.61 (Rank 5)
SMOTE-ENN	5.37 (Rank 3)	5.33 (Rank 3)
SMOTE-TL	5.39 (Rank 4)	5.17 (Rank 2)
SMOTE	5.70 (Rank 5)	5.42 (Rank 4)
Adasyn	6.14 (Rank 6)	5.72 (Rank 6)
NCL	6.62 (Rank 7)	7.33 (Rank 8)
BSMOTE	7.10 (Rank 8)	7.49 (Rank 9)
SD2	7.64 (Rank 9)	7.68 (Rank 10)
ROS	7.97 (Rank 10)	7.32 (Rank 7)
SD1	8.56 (Rank 11)	8.73 (Rank 11)
CNN	9.70 (Rank 12)	9.41 (Rank 12)
No-Resampling	11.51 (Rank 13)	11.54 (Rank 13)

Table 28
Wilcoxon Signed Rank Test Results.

Resampling techniques pair	Test Statistics with respect to GM	Test Statistics with respect to BL
Safe-SMOTE – Adasyn	Sig (p-value = 0.000)	Sig (p-value = 0.000)
Safe- SMOTE vs. BSMOTE	Sig (p-value = 0.000)	Sig (p-value = 0.000)
Safe-SMOTE vs. SMOTE-ENN	Sig (p-value = 0.000)	Sig (p-value = 0.000)
Safe-SMOTE vs. SMOTE	Sig (p-value = 0.000)	Sig (p-value = 0.000)
Safe-SMOTE vs. SMOTE-TL	Sig (p-value = 0.000)	Sig (p-value = 0.000)
Safe-SMOTE vs. SD1	Sig (p-value = 0.000)	Sig (p-value = 0.000)
Safe-SMOTE vs. SD2	Sig (p-value = 0.000)	Sig (p-value = 0.000)
Safe-SMOTE vs. ROS	Sig (p-value = 0.000)	Sig (p-value = 0.000)
Safe-SMOTE vs. CNN	Sig (p-value = 0.000)	Sig (p-value = 0.000)
Safe-SMOTE vs. NCL	Sig (p-value = 0.000)	Sig (p-value = 0.000)
Safe-SMOTE vs. RUS	NotSig (p-value = 0.878)	NotSig (p-value = 0.005)
Safe-SMOTE vs. No-Resampling	Sig (p-value = 0.000)	Sig (p-value = 0.000)

conducted the Friedman test for GM and BL after applying all data resampling techniques. For example, to evaluate the performance of the X technique, we extracted the GM and BL values of class maintainability prediction models developed on all eight datasets after applying all data resampling techniques. The Friedman test analysis was carried out at a significance level of α = 0.05 with 27 degrees of freedom (28 classification techniques). The Friedman test assesses the hypothesis that the performance of different ML and SB techniques on all datasets is not different from each other. The mean ranks obtained after the Friedman test in terms of GM and BL values of all ML and SB techniques (median GM and BL on ten runs of SB techniques) on all eight datasets and all data resampling techniques are depicted in Table 29. The p-values obtained after conducting the test was 0.000 both for GM and BL that indicate the results are significant. It yields the rejection of the null hypothesis and leads to the conclusion that the performance of different ML and SB techniques on all datasets are different from each other. As shown in Table 29, CHC technique has obtained best rank in terms of GM and second-best rank in terms of BL whereas LRR technique has obtained best rank in terms of GM and second-best rank in terms of BL. It is to be noted that CHC, SGA, PBIL, SSMA, GAssist-Int, GAssist-ADI, GGA, MLPCS, and CSO are the top ten classification techniques as per the Friedman test ranking in terms of GM and BL and all these techniques are SM techniques. This implies that these techniques can be used for developing prediction models for class maintainability across different software systems. The worst performing techniques in terms of GM and BL are RIPPER, AB, BNGE, RISE, KNN, KSTAR, EACH, PART, and UCS. The PART technique has obtained the worst rank.

As the results of the Friedman test were significant, we further carry out post-hoc analysis with the help of Wilcoxon signed-rank test with Bonferroni correction where we have adjusted the level of significance to α = 0.05/27 = 0.001, as we have evaluated 27 pairs of classification techniques. As per the Friedman test ranking the most accurate method for predicting the class maintainability was LRR in terms of GMvalues. Also, according to the Wilcoxon signed-rank test, LRR is significantly superior to 20 out of 28 classification techniques. Further LRR is not significantly different in the performance in terms of GM (α = 0.001) for 7 out of 28 classification techniques namely-CHC, SGA, PBIL, SSMA, GAssist-ADI, GAssist-Int, and CPSO (Table 30). CHC technique was the best technique for predicting the class maintainability in terms of BL values. Also, CHC is found superior to 23 out of 28 classification techniques according to the Wilcoxon signed-rank test. CHC is not significantly different in the performance in terms of BL for 5 out of 28 classification techniques namely-LRR, SSMA, GAssist-ADI, GAssist-Int, and CPSO. Also, CHC, LRR, SGA, PBIL, SSMA, GAssist-Int, GAssist-ADI, GGA, MLPCS, and CSO are the top ten classification techniques as per Friedman test. All of these are SB techniques except LRR.

7. Empirical validity threats

We analyze various threats to the empirical validity of the study in this section. The correct way of measuring the independent and dependent variables for developing the model for a predictive task ensures the construct validity. The independent variables used in this study are prominent OO metrics used in various studies to build software quality models and the dependent variable class maintainability measured in terms of changes in lines of code in multiple studies in the literature. Therefore, for variables, there is no threat to construct validity. Selection of classification techniques to develop the models could also be a possible threat to construct validity. However, we have reduced this threat by exploring a wide range of ML and SB techniques. An accurate analysis of the results of the study to derive the conclusions ensures the conclusion validity. The class maintainability prediction models are built by ten-fold cross-validation, and the results are analyzed statistically. Hence, there the threat to conclusion validity does not exist in this study. There is a change in the original ratio of minority and majority class datapoints due to data resampling. Due to the difference in the data distribution, the causal relationship between the independent and dependent might have altered that

Table 29
Friedman Test Results for Performance of Classification Techniques used in the Study

Classification Techniques	Category	Mean rank with respect to BL	Classification Techniques	Category	Mean rank with respect to GM
CHC	SB	7.79	LRR	ML	7.57
LRR	ML	7.80	CHC	SB	8.06
SGA	SB	10.51	GAINT	SB	10.10
PBIL	SB	10.51	GAssist-ADI	SB	10.50
SSMA	SB	10.51	SSMA	SB	10.57
GAssist-Int	SB	10.66	PBIL	SB	10.72
GAssist-ADI	SB	10.68	SGA	SB	10.94
GGA	SB	11.38	GGA	SB	11.73
MPLCS	SB	11.91	MPLCS	SB	11.77
CPSO	SB	12.05	CPSO	SB	12.85
LWPSO	SB	13.13	XCS	SB	13.19
SLIPPER	ML	13.41	SLIPPER	ML	13.56
XCS	SB	13.63	CHIRW	ML	13.86
BG	ML	14.65	BG	ML	13.93
IGA	SB	14.72	LWPSO	SB	14.63
PUBLIC	ML	14.84	PUBLIC	ML	14.87
CHIRW	ML	14.95	BIOHEL	SB	14.91
C.45	ML	15.04	C.45	ML	15.05
BIOHEL	SB	15.38	IGA	SB	15.13
RIPPER	ML	15.59	RIPPER	ML	15.86
AB	ML	15.94	AB	ML	15.98
BNGE	ML	16.87	BNGE	ML	16.68
RISE	ML	17.46	RISE	ML	17.55
KNN	ML	17.61	KNN	ML	17.83
KSATR	ML	20.25	KSATR	ML	19.40
EACH	ML	21.14	EACH	ML	21.18
PART	ML	22.38	PART	ML	22.57
UCS	SB	25.25	UCS	SB	25.01

Table 30
Wilcoxon Signed Rank Test Results for Performance of Classification Techniques used in the Study

Classification Techniques pair	Test statistics with respect to GM	Classification Techniques pair	Test statistics with respect to BL
LRR-CHC	NotSig (p-value = 0.324)	CHC-LRR	NotSig (p-value = 0.423)
LRR-SGA	NotSig (p-value = 0.001)	CHC-SGA	Sig (p-value = 0.000)
LRR-PBIL	NotSig (p-value = 0.002)	CHC-PBIL	Sig (p-value = 0.000)
LRR-SSMA	NotSig (p-value = 0.005)	CHC-SSMA	NotSig (p-value = 0.001)
LRR-GAssist-Int	NotSig (p-value = 0.008)	CHC-GAssist-Int	NotSig (p-value = 0.001)
LRR-GAssist-ADI	NotSig (p-value = 0.003)	CHC-GAssist-ADI	NotSig (p-value = 0.001)
LRR-GGA	Sig (p-value = 0.000)	CHC-GGA	Sig (p-value = 0.000)
LRR-MPLCS	Sig (p-value = 0.000)	CHC-MPLCS	Sig (p-value = 0.000)
LRR-CPSO	NotSig (p-value = 0.010)	CHC-CPSO	NotSig (p- value = 0.011)
LRR-LWPSO	Sig (p-value = 0.000)	CHC-LWPSO	Sig (p-value = 0.000)
LRR-SLIPPER	Sig (p-value = 0.000)	CHC-SLIPPER	Sig (p-value = 0.000)
LRR-XCS	Sig (p-value = 0.000)	CHC-XCS	Sig (p-value = 0.000)
LRR-BG	Sig (p-value = 0.000)	CHC-BG	Sig (p-value = 0.000)
LRR-IGA	Sig (p-value = 0.000)	CHC-IGA	Sig (p-value = 0.000)
LRR-PUBLIC	Sig (p-value = 0.000)	CHC-PUBLIC	Sig (p-value = 0.000)
LRR-CHIRW	Sig (p-value = 0.000)	CHC-CHIRW	Sig (p-value = 0.000)
LRR-C4.5	Sig (p-value = 0.000)	CHC-C4.5	Sig (p-value = 0.000)
LRR-BIOHEL	Sig (p-value = 0.000)	CHC-BIOHEL	Sig (p-value = 0.000)
LRR-RIPPER	Sig (p-value = 0.000)	CHC-RIPPER	Sig (p-value = 0.000)
LRR-AB	Sig (p-value = 0.000)	CHC-AB	Sig (p-value = 0.000)
LRR-BNGE	Sig (p-value = 0.000)	CHC-BNGE	Sig (p-value = 0.000)
LRR-RISE	Sig (p-value = 0.000)	CHC-RISE	Sig (p-value = 0.000)
LRR-KNN	Sig (p-value = 0.000)	CHC-KNN	Sig (p-value = 0.000)
LRR-KSATR	Sig (p-value = 0.000)	CHC-KSTAR	Sig (p-value = 0.000)
LRR-EACH	Sig (p-value = 0.000)	CHC-EACH	Sig (p-value = 0.000)
LRR-PART	Sig (p-value = 0.000)	CHC-PART	Sig (p-value = 0.000)
LRR-UCS	Sig (p-value = 0.000)	CHC-UCS	Sig (p-value = 0.000)

could be a possible threat to internal validity. However, we have used stable performance metrics GM and BL to assess our results that reduce the risk to internal validity. The extent to which the results of the empirical study are generalizable and applicable refers to as external validity. The datasets used in this empirical investigation are the representations of the open-source software Apache, which is written in Java programming. Therefore, there may threat that the results would not hold equally well for proprietary software or software system written in programming languages other than Java.

8. Conclusions and future work

The study deals with developing efficient and effective models for predicting software classes having low maintainability. These classes require high maintenance effort and more resources in the software maintenance phase, therefore, increase the overall cost of the software development. Early prediction of such classes helps in better designing and implementation of these classes. Generally, for a software project, such classes are very few, causing imbalanced data to develop prediction models to predict these classes. This study explores and assesses the performance of different data resampling techniques to handle imbalanced data. After, data balancing, the study develops class maintainability prediction models by 28 classification techniques, including 14 ML and 14 SB techniques. For performance assessment of the developed models, the stable performance metrics, GM and BL are used. The study also carried out extensive statistical analysis on the results with the help of the Friedman test and Wilcoxon signed-rank test. The results of the study confirmed that Safe-SMOTE is competent technique to handle the imbalanced datasets.

The performance of RUS is found comparable with Safe-SMOTE. After, data resampling the performance analysis of ML, and SB techniques was carried out. It was found that CHC, LRR, SGA, PBIL, SSMA, GAssist-ADI, GAssist-Int, and CPSO are the competent techniques for predicting low maintainability software classes accurately. The worst performing techniques in terms of GM and BL are RIPPER, AB, BNGE, RISE, KNN, KSTAR, EACH, PART, and UCS. All of these are ML techniques except UCS, which is SB technique. As, this study is carried out on eight Apache software packages, in the future, we plan to extend this investigation on more datasets from different applications domains and programming languages. Also, in the future, we plan to continue this study using hybridized techniques.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Y. Ahn, J. Suh, S. Kim, H. Kim, The software maintenance project effort estimation model based on function points, J. Softw. Maintenance Evol. Res. Pract. 15 (2) (2003) 71–85.
[2] K. Erdil, E. Finn, K. Keating, J. Meattle, S. Park, D. Yoon, Software maintenance as part of the software life cycle, Comp180: Software Engineering Project, 2003, pp. 1–49.
[3] L.C. Briand, C. Bunse, J.W. Daly, C. Differding, An experimental comparison of the maintainability of object-oriented and structured design documents, Emp. Softw. Eng. 2 (3) (1997) 291–312.
[4] IEEE, IEEE standard glossary of software engineering terminology, IEEE Std 610.12-1990, Institute of Electrical and Electronics Engineering, 1990.
[5] W. Li, S. Henry, Object-oriented metrics that predict maintainability, J. Syst. Softw. 23 (2) (1993) 111–122.
[6] N.E. Fenton, S.L. Pfleeger, Software Metrics: A Rigorous and Practical Approach, 2nd ed., PWS Publishing Company: ITP, Boston, MA, 1997.
[7] S. Morasca, A probability-based approach for measuring external attributes of software artifacts, in: Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement (pp. 44-55). IEEE Computer Society, 2009.
[8] J. Al Dallal, Object-oriented class maintainability prediction using internal quality attributes, Inf. Softw. Technol. 55 (11) (2013) 2028–2048.
[9] Y. Lee, K.H. Chang, Reusability and maintainability metrics for object-oriented software, in: Proceedings of the 38th annual on Southeast regional conference (pp. 88-94). ACM, 2000.
[10] L.C. Briand, S. Morasca, V.R. Basili, Measuring and assessing maintainability at the end of high level design. In 1993 Conference on Software Maintenance(pp. 88-87). IEEE, 1993.

[11] M. Dagpinar, J.H. Jahnke, Predicting maintainability with object-oriented metrics-an empirical comparison, in: 10th Working Conference on Reverse Engineering, 2003. WCRE 2003. Proceedings. (pp. 155-164). IEEE.
[12] Y. Zhou, H. Leung, Predicting object-oriented software maintainability using multivariate adaptive regression splines, J. Syst. Softw. 80 (8) (2007) 1349–1361.
[13] L.J. Wang, X.X. Hu, Z.Y. Ning, W.H. Ke, Predicting object-oriented software maintainability using projection pursuit regression, in: 2009 First International Conference on Information Science and Engineering, IEEE, 2009, pp. 3827–3830.
[14] K.K. Aggarwal, Y. Singh, A. Kaur, R. Malhotra, Application of artificial neural network for predicting maintainability using object-oriented metrics, Trans. Eng. Comput. Technol. 15 (2006) 285–289.
[15] R. Malhotra, K. Lata, An Exploratory Study for Predicting Maintenance Effort using Hybridized Techniques, in: Proceedings of the 10th Innovations in Software Engineering Conference. ACM, 2017, pp. 26–33.
[16] L. Song, D. Li, X. Zeng, Y. Wu, L. Guo, Q. Zou, nDNA-prot: identification of DNA-binding proteins based on unbalanced classification, BMC Bioinf. 15 (1) (2014) 298.
[17] A.C. Bahnsen, A. Stojanovic, D. Aouada, B. Ottersten, Cost sensitive credit card fraud detection using Bayes minimum risk, 2013 12th international conference on machine learning and applications, vol. 1, IEEE, 2013, pp. 333–338.
[18] A. Abbasi, H. Chen, A comparison of fraud cues and classification methods for fake escrow website detection, Inf. Technol. Manage. 10 (2–3) (2009) 83–101.
[19] D. Rodriguez, I. Herraiz, R. Harrison, J. Dolado, J.C. Riquelme, Preliminary comparison of techniques for dealing with imbalance in software defect prediction, in: Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering, ACM, 2014, p. 43.
[20] M. Tan, L. Tan, S. Dara, C. Mayeux, Online defect prediction for imbalanced data, 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering, vol. 2, IEEE, 2015, pp. 99–108.
[21] R. Malhotra, M. Khanna, An empirical study for software change prediction using imbalanced data, Emp. Softw. Eng. 22 (6) (2017) 2806–2851.
[22] G. Haixiang, L. Yijing, J. Shang, G. Mingyun, H. Yuanyue, G. Bing, Learning from class-imbalanced data: review of methods and applications, Expert Syst. Appl. 73 (2017) 220–239.
[23] R. Malhotra, M. Khanna, An exploratory study for software change prediction in object-oriented systems using hybridized techniques, Automated Softw. Eng. 24 (3) (2017) 673–717.
[24] S. Hosseini, B. Turhan, M. Mäntylä, A benchmark study on the effectiveness of search-based data selection and feature selection for cross project defect prediction, Inf. Softw. Technol. 95 (2018) 296–312.
[25] R. Malhotra, M. Khanna, R.R. Raje, On the application of search-based techniques for software engineering predictive modeling: A systematic review and future directions, Swarm Evol. Comput. 32 (2017) 85–109.
[26] T.M. Khoshgoftaar, R.M. Szabo, Improving Code Churn Predictions During the System Test and Maintenance Phases, ICSM 94 (1994) 58–67.
[27] M.M.T. Thwin, T.S. Quah, Application of neural networks for software quality prediction using object-oriented metrics, J. Syst. Softw. 76 (2) (2005) 147–156.
[28] M.O. Elish, K.O. Elish, Application of treenet in predicting object-oriented software maintainability: A comparative study, in: 2009 13th European Conference on Software Maintenance and Reengineering, IEEE, 2009, pp. 69–78.
[29] S.O. Olatunji, Z. Rasheed, K.A. Sattar, A.M. Al-Mana, M. Alshayeb, E.A. El-Sebakhy, Extreme learning machine as maintainability prediction model for object-oriented software systems, J. Comput. 2 (8) (2010) 49–56.
[30] W. Zhang, L. Huang, V. Ng, J. Ge, SMPLearner: learning to predict software maintainability, Automated Softw. Eng. 22 (1) (2015) 111–141.
[31] L. Kumar, S.K. Rath, Neuro–genetic approach for predicting maintainability using Chidamber and Kemerer software metrics suite, in: Recent Advances in Information and Communication Technology 2015, Springer, Cham, 2015, pp. 31–40.
[32] H. Alsolai, M. Roper, A Systematic Literature Review of Machine Learning Techniques for Software Maintainability Prediction, Information and Software Technology, 2019, p. 106214.
[33] H. Yu, J. Ni, Y. Dan, S. Xu, Mining and integrating reliable decision rules for imbalanced cancer gene expression data sets, Tsinghua Sci. Technol. 17 (6) (2012) 666–673.
[34] T. Choeikiwong, P. Vateekul, Software defect prediction in imbalanced data sets using unbiased support vector machine, in: Information Science and Applications, Springer, Berlin, Heidelberg, 2015, pp. 923–931.
[35] T.M. Khoshgoftaar, K. Gao, N. Seliya, Attribute selection and imbalanced data: Problems in software defect prediction, 2010 22nd IEEE International Conference on Tools with Artificial Intelligence, vol. 1, IEEE, 2010, pp. 137–144.
[36] I.H. Laradji, M. Alshayeb, L. Ghouti, Software defect prediction using ensemble learning on selected features, Inf. Softw. Technol. 58 (2015) 388–402.
[37] M.J. Siers, M.Z. Islam, Software defect prediction using a cost sensitive decision forest and voting, and a potential solution to the class imbalance problem, Information Syst. 51 (2015) 62–71.
[38] L. Pelayo, S. Dick, Applying novel resampling strategies to software defect prediction, in: NAFIPS 2007-2007 Annual Meeting of the North American Fuzzy Information Processing Society. IEEE, 2007, pp. 69–72.
[39] T. Menzies, A. Dekhtyar, J. Distefano, J. Greenwald, Problems with Precision: A Response to“ comments on 'data mining static code attributes to learn defect predictors'”, IEEE Trans. Software Eng. 33 (9) (2007) 637–640.
[40] N. Seliya, T.M. Khoshgoftaar, J. Van Hulse, Predicting faults in high assurance software, in: 2010 IEEE 12th International Symposium on High Assurance Systems Engineering, IEEE, 2010, pp. 26–34.
[41] C. Seiffert, T. Khoshgoftaar, J. Van Hulse, Improving software quality predictions with data sampling and boosting, IEEE Trans. Syst. Man Cybern. A, Syst. Humans 39 (6) (2009) 1283–1294.
[42] http://gromit.iiar.pwr.wroc.pl/p_inf/ ckjm/metric.html.
[43] D.W. Zimmerman, B.D. Zumbo, Relative power of the Wilcoxon test, the Friedman test, and repeated-measures ANOVA on ranks, J. Exp. Educ. 62 (1) (1993) 75–86.
[44] R. Malhotra, N. Pritam, K. Nagpal, P. Upmanyu, Defect collection and reporting system for git based open source software, in: 2014 International Conference on Data Mining and Intelligent Computing (ICDMIC), IEEE, 2014, pp. 1–7.
[45] https://www.cs.waikato.ac.nz/ml/weka/.
[46] M.A. Hall, Correlation-based feature selection for machine learning, 1999.
[47] M.A. Hall, G. Holmes, Benchmarking attribute selection techniques for discrete class data mining, IEEE Trans. Knowledge Data Eng. 15 (6) (2003) 1437–1447.
[48] R. Malhotra, A systematic review of machine learning techniques for software fault prediction, Appl. Soft Comput. 27 (2015) 504–518.
[49] E. Arisholm, L.C. Briand, E.B. Johannessen, A systematic and comprehensive investigation of methods to build and evaluate fault prediction models, J. Syst. Softw. 83 (1) (2010) 2–17.
[50] A.B. De Carvalho, A. Pozo, S.R. Vergilio, A symbolic fault-prediction model based on multiobjective particle swarm optimization, J. Syst. Softw. 83 (5) (2010) 868–882.
[51] R. Malhotra, M. Khanna, Investigation of relationship between object-oriented metrics and change proneness, Int. J. Mach. Learn. Cybern. 4 (4) (2013) 273–286.
[52] N.V. Chawla, N. Japkowicz, A. Kotcz, Special issue on learning from imbalanced data sets, ACM Sigkdd Explorations Newsletter 6 (1) (2004) 1–6.
[53] H. He, Y. Bai, E.A. Garcia, S. Li, ADASYN: adaptive synthetic sampling approach for imbalanced learning, in: 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence). IEEE, 2008, pp. 1322–1328.
[54] G.E. Batista, R.C. Prati, M.C. Monard, A study of the behavior of several methods for balancing machine learning training data, ACM SIGKDD Explorations Newsletter 6 (1) (2004) 20–29.
[55] N.V. Chawla, K.W. Bowyer, L.O. Hall, W.P. Kegelmeyer, SMOTE: synthetic minority over-sampling technique, J. Artif. Intell. Res. 16 (2002) 321–357.
[56] H. Han, W.Y. Wang, B.H. Mao, Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning, in: International conference on intelligent computing, Springer, Berlin, Heidelberg, 2005, pp. 878–887.
[57] Bunkhumpornpat, C., Sinapiromsaran, K., & Lursinsap, C. (2009, April). Safe-level-smote: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem. In Pacific-Asia conference on knowledge discovery and data mining (pp. 475-482). Springer, Berlin, Heidelberg.
[58] J. Stefanowski, S. Wilk, Selective pre-processing of imbalanced data for improving classification performance, in: International conference on intelligent computing, Springer, Berlin, Heidelberg, 2008, pp. 283–292.
[59] K. Napierała, J. Stefanowski, S. Wilk, Learning from imbalanced data in presence of noisy and borderline examples, in: International Conference on Rough Sets and Current Trends in Computing. Springer, Berlin, Heidelberg, 2010, pp. 158–167.
[60] J. Laurikkala, Improving identification of difficult small classes by balancing class distribution, in: Conference on Artificial Intelligence in Medicine in Europe, Springer, Berlin, Heidelberg, 2001, pp. 63–66.
[61] www.keel.es.

Ruchika Malhotra is Associate Head and Associate Professor in the Discipline of Software Engineering, Department of Computer Science and Engineering, Delhi Technological University (formerly Delhi College of Engineering), Delhi, India. She has been awarded the prestigious UGC Raman Postdoctoral Fellowship by the Indian government for pursuing postdoctoral research from Department of Computer and Information Science, Indiana University-Purdue University, Indianapolis, Indiana, USA. She was an assistant professor at the University School of Information Technology, Guru Gobind Singh Indraprastha University, Delhi, India. She received her masters and doctorate degree in software engineering from the University School of Information Technology, Guru Gobind Singh Indraprastha University, Delhi, India. She has received IBM Faculty Award 2013. She has received Best Presenter Award in Workshop on Search Based Software Testing, ICSE, 2014, Hyderabad, India. Her h-index is 26 as reported by Google Scholar. She can be contacted by email at ruchikamalhotra2004@yahoo.com

Kusum Lata is currently working as Assistant Professor in University School of Management and Entrepreneurship, Delhi Technological University and pursuing her doctoral degree from Delhi Technological University. She completed her master's degree in Computer Technology and Applications in 2010 from the Delhi College of Engineering, University of Delhi, India. She received her bachelor degree in Computer Science and Engineering in 2003 from Beant College of Engineering and Technology, Punjab Technical University, India. Her research interests are in software quality improvement, applications of machine learning techniques in maintainability prediction, and validation of software metrics. She can be contacted by email at er.kusum82@gmail.com