Empir Software Eng (2017) 22:2806–2851
DOI 10.1007/s10664-016-9488-7

An empirical study for software change prediction using imbalanced data

Ruchika Malhotra[1] · **Megha Khanna**[1]

Published online: 5 January 2017

Abstract Software change prediction is crucial in order to efficiently plan resource allocation during testing and maintenance phases of a software. Moreover, correct identification of change-prone classes in the early phases of software development life cycle helps in developing cost-effective, good quality and maintainable software. An effective software change prediction model should equally recognize change-prone and not change-prone classes with high accuracy. However, this is not the case as software practitioners often have to deal with imbalanced data sets where instances of one type of class is much higher than the other type. In such a scenario, the minority classes are not predicted with much accuracy leading to strategic losses. This study evaluates a number of techniques for handling imbalanced data sets using various data sampling methods and MetaCost learners on six open-source data sets. The results of the study advocate the use of resample with replacement sampling method for effective imbalanced learning.

Keywords Change proneness · Data sampling · Empirical validation · Imbalanced learning · MetaCost learners · Object-oriented metrics

1 Introduction

Software change prediction involves determination of classes which are prone to change in the future releases of the software product by taking into account various design

Communicated by: Jeffrey Carver

✉ Ruchika Malhotra
ruchikamalhotra2004@yahoo.com

Megha Khanna
meghakhanna86@gmail.com

[1] Delhi Technological University, Delhi, India

metrics. These metrics represent various Object-Oriented (OO) software characteristics such as coupling, cohesion, size etc. Identification of change prone classes is essential as these classes require more attention during maintenance and testing phase of the software as they are probable sources of defects and advancements (Eski and Buzluca 2011; Malhotra and Khanna 2013) i.e. they are sources of defects based changes, advancement bases changes as well as maintenance based changes. Proper allocation of resources to such classes helps in improving the quality of the software product as stringent verification activities such as inspections and reviews can be performed on such classes in the early phases of software development so that defects and probable changes can be identified as early as possible. Such a practice would help in easy management of changes and efficient correction of defects in the initial phases of the software life cycle. However, if these defects and changes are propagated to later phases of the software product life cycle, they become costlier to correct and manage. Furthermore, early determination of change prone classes facilitates preventive design measures, which may be taken by software professionals so that future errors may be avoided in these classes (Koru and Tian 2005; Koru and Liu 2007). Also, activities like refactoring can be directed on change-prone classes so that these classes are better adept at handling future changes and the effect of changes made to these classes may be localized (Elish and Al-Khiaty 2013). Thus, development of models for effective determination of change prone classes is important in order to release and maintain good quality software products at optimum costs.

A number of studies in the past (Bieman et al. 2001; Koru and Tian 2005; Eski and Buzluca 2011; Lu et al. 2012; Elish and Al-Khiaty 2013; Malhotra and Khanna 2013) have validated statistical as well as Machine-Learning (ML) methods for determination of change prone classes on various commercial and open-source data sets. In order to develop a useful change prediction model, a learning technique requires an efficient training data set, which have sufficient number of both change-prone and not change-prone classes so that the model can effectively learn to identify them. However, in real world a number of data sets are imbalanced in nature i.e. a majority of classes belong to a particular category, with very few instances of the other category, which is generally of more interest (change prone in our case). Learning from such imbalanced data sets leads to higher misclassifications for the minority class. This is because of lack of information available about the minority class. Such models are rarely of any practical use as the important change-prone classes are neglected and may not be properly identified. This would lead to improper testing and maintenance plans as sufficient resources would not be allotted to the unidentified change prone classes leading to poor quality software products. Moreover, such products would be costlier to maintain and manage and may not be completed under tight schedule deadlines. This study deals with two specific approaches for efficiently learning from imbalanced data sets while developing change prediction models: data sampling approaches and cost-sensitive classification.

(i) Data Sampling Approach: This approach involves modification of training data in such a manner so as to provide adequate number of training instances for both change-prone and not change-prone classes. For example, a researcher can oversample the number of minority instances to increase the ratio of minority examples and improve the classifier's performance (Chawla et al. 2002; Lopez et al. 2013).

(ii) Cost-Sensitive Classification: This approach involves giving appropriate weights to various misclassification errors so that the model learns in such a fashion that it reduces the cost of misclassifications (Domingos 1999; Lopez et al. 2013).

Apart from these methodologies, this study also adopts stable performance metrics which should be evaluated when we are learning through imbalanced data sets. Thus, the research questions (RQ) explored in the study are as follows:

RQ1a: Does the performance of different ML methods in this study significantly improve by using various sampling approaches for Imbalanced Learning Problem (ILP) in change prediction?
RQ1b: Which sampling approach performs best amongst different sampling techniques analysed in the study?
RQ2: What is the effectiveness of different MetaCost learners (Domingos 1999) for improving learning through imbalanced data?
RQ3: What is comparative performance of the best sampling approach and MetaCost learner for ILP?

In order to evaluate the above RQ's we empirically evaluate six widely used open-source data sets using several ML methods. The ILP was addressed by using three data sampling approaches namely, Resampling with replacement, Synthetic Minority Oversampling Technique (SMOTE), and Spread Subsample and by using MetaCost learners for cost-sensitive learning. Furthermore, the study assesses the performance using several stable performance measures like G-mean, Receiver Operating Characteristic (ROC) analysis and balance. The study also compares these stable performance metrics with traditional metrics such as "accuracy", "recall" and "precision". This study develops change prediction models using ten-fold cross validation and inter-release validation. Moreover, the study performs statistical analysis in order to compare various methods for handling ILP and evaluate their effectiveness. The results of the study advocate the use of the above mentioned approaches for developing models using imbalanced data. Furthermore, the study supports the use of resample with replacement sampling method as the most effective approach amongst the evaluated approaches.

The organization of the study is as follows: Section 2 briefly overviews the related literature and Section 3 discusses the ILP in detail. Section 4 discusses the various design aspects of the study and Section 5 describes the experimental design. Section 6 states the research methodology. Section 7 states and analyses the results of the study using ten-fold cross validation and Section 8 discusses the results specific to each RQ using inter-release validation. Section 9 discusses the various threats to validity of the study and Section 10 states the conclusions of the study and provides pointers for future work.

2 Related Work

This section discusses the related work of the study and is organized into three sub-sections. The first section states the studies related to change prediction. The second section reports the various approaches for solving the ILP. The third section states the studies related to ILP on various real-life and software engineering applications especially software defect prediction and change prediction.

2.1 Change Prediction Studies

There are multiple reasons for a change in a software which include existence of defects, change in requirements or upgradation of software. Various researchers in the past have developed models for predicting change prone nature of a class. A study by Bieman et al. (2001) validated an OO system which was commercial in nature to evaluate the relationship between design structure and software changes. In order to determine the change prone classes, the study used design patterns, size metrics and inheritance metrics. Studies by Koru and Tian (2005) and Koru and Liu (2007) use various OO metrics which are indicators of inheritance, size, coupling and complexity attributes of a software on two widely used open-source data sets (KOffice and Mozilla) in order to ascertain change-prone classes. A study by Zhou et al. (2009) validated the confounding effect of size of a class on the relationship between OO metrics and change-prone nature of a class. The study used two releases of Eclipse software, which is a framework of tools for building and deploying software projects. Lu et al. (2012), Eski and Buzluca (2011) and Malhotra and Khanna (2013) evaluated a number of open-source Java projects for ascertaining the nature of change prone classes using various OO metrics which represent size, cohesion, coupling and inheritance attributes of a software. A recent study by Elish and Al-Khiaty (2013) successfully used evolution-based metrics along with OO metrics for determining the change prone nature of a class on two open-source software data sets. A study by Romano and Pinzger (2011) predicted change-prone Java interfaces on Hibernate and Eclipse open-source software data sets using cohesion metrics and other OO metrics. Another study which successfully evaluated change prone classes using OO metrics on two large open source software projects was conducted by Giger et al. (2012). These studies confirm the effective use of change prediction models in various commercial and open source software data sets. Software practitioners can use these models for efficient resource allocation of limited project resources such as its budget, effort and time to the identified change prone classes so that good quality software products are delivered. However, use of imbalanced data sets may pose problems in developing effective change prediction models and the next study discusses various approaches for handling the ILP.

2.2 Approaches for Handling ILP

It is important for ILP to be effectively handled in order to provide efficient learning for model development. A number of issues are associated with ILP. Visa and Ralescu (2005) briefly overview the field of imbalanced learning to discuss the nature and issues associated with imbalanced data. They attribute the ineffective performance of various learning methods to three primary reasons i.e. accuracy, class distribution and error costs. Studies by He and Garcia (2009), Weiss (2004), Zhang and Li (2011) and Lopez et al. (2013) discuss various state of the art methods for developing effective models using imbalanced data. A broad categorization of the various methods suggested by these studies is as follows: (i) use of sampling methods, (ii) use of cost sensitive methods, (iii) use of kernel-based and active learning methods, (iv) use of ensemble learners, (v) use of appropriate evaluation metrics (Weng and Poon 2008; Jeni et al. 2013; Bekkar et al. 2013), (vi) use of knowledge/human interaction, (vii) data segmentation, (viii) use of non-greedy techniques for searching, (ix) use of an effective inductive bias and (x) use of additional methods such as one-class learning methods or novelty detection methods amongst other additional methods. It may be noted that there are certain classification methods that do not assume that the data is balanced is nature. One such method is Mahalanobis-

Taguchi method (Hirohisa et al. 2006; Su and Hsiao 2007), which has been used for developing classification models on imbalanced data sets in quality engineering field. However, these methods have been rarely use in the software engineering field. A prime reason for such a trend is the difficulty in determination of a suitable threshold to conduct efficient classification process while learning from one class samples (Su and Hsiao 2007).

2.3 ILP on Various Real-Life and Software Engineering Problems

A number of various real life applications face ILP, which include sentiment analysis (Xu et al. 2015), fraud detection (Phua et al. 2004), video mining (Apandi et al. 2011), text mining (Munkhdalai et al. 2015), churn prediction (Bekkar et al. 2013) and various bioinformatics applications (Yang et al. 2014) amongst many others. However, in terms of software engineering problems, defect prediction using imbalanced learning data is widely explored by a number of researchers. A study by Wang and Yao (2013) investigated threshold mining, resampling techniques and ensembles techniques for software defect prediction using imbalanced data sets. A number of other studies including Shatnawi (2012), Kamei et al. (2007), Seiffert et al. (2014), Liu et al. (2006) have explored the use of different sampling methods in order to improve the performance of software defect prediction models.

Seliya and Khoshgoftaar (2011) analysed different cost-sensitive learning techniques using decision trees for developing defect prediction models on software data sets. They also took into account misclassification cost as an important parameter for model training. Studies by Weiss et al. (2007), Galar et al. (2012) and Rodriguez et al. (2014) analysed cost-sensitive, sampling and ensemble learning approaches and compared them for developing effective software defect prediction models using imbalanced data. A study by Gao et al. (2015) suggest the use of feature selection along with sampling approaches to learn effectively from imbalanced defect data sets.

Though a number of studies assessed and improved the development of software defect prediction models, only one study by Tan et al. (2015) evaluated change classification models, which is a related area of defect prediction. This study assessed the comparative performance of resampling techniques and updateable classification method for imbalanced learning and ascertained that the performance of developed models improve with the use of these techniques.

This study uses two methods (use of sampling approaches and cost-sensitive methods) for handling ILP. To the best of author's knowledge, this study is a pioneer in statistically evaluating the performance of sampling methods and MetaCost learners on imbalanced data sets for developing change prediction models. Moreover, the study uses effective performance evaluators as discussed in literature for evaluating models created using imbalanced data. Also, the models are compared using certain traditional metrics, which helps in comparing and assessing the imbalanced learning problem from various aspects.

3 The Imbalanced Learning Problem

Imbalanced Learning problem is well recognized in the literature where a data set has large mismatch between the number of instances of different categories of classes. This study deals with binary class imbalance problem where a data set has only two categories of classes namely change prone and not change prone. We generally find that change-prone classes are

under-represented in data sets as software data sets follow Pareto principle. This means that bulk of changes and defects in a data set originate from only 20% of classes or modules (Koru and Tian 2005). Thus, it is important to develop software quality models to identify change prone classes as these classes should be efficiently designed and rigorously tested to improve the quality of the software.

An effective classification model should be able to detect both change-prone and not change-prone classes with high accuracies. However, due to imbalanced nature of the training data, it is often the case when not change prone classes are recognized with higher accuracies but change prone classes are not that well recognized giving lower accuracy rates. However, since the datasets are highly imbalanced, this lower recognition of change prone classes is ignored and the classifiers exhibit good overall accuracy rates. Such classifiers are imbalanced but are still termed as good classification models. However, this scenario is quite unfavourable and may lead to erroneous judgements leading to high losses and bad reputation for the software organization.

Let us understand the scenario with the help of an example. Suppose we have a data sets with 500 data points, where each data point is a collection of OO metrics and the change prone or not change prone nature of a class. The goal is to develop an accurate and unbiased model which detects both change prone and not change prone classes correctly. However, the data set is imbalanced with only 10% of change prone classes and the other not change prone in nature. A model trained on such a data set would show correct and accurate predictions for not change prone classes (close to 100%) but very low accurate predictions for change prone classes ranging from 0 to 10% (He and Garcia 2009). Thus, assuming 10% accuracy for change prone classes only 5 out of 50 change prone classes are accurately predicted. Such low prediction accuracy for change prone classes is not advisable as it would lead to a bad quality software product. This is because change prone classes require allocation of sufficient resources so that they can be effectively designed, scrutinized, verified and tested. These classes should be properly handled in order to prevent occurrences of defects and changes in future versions of the software. A low accuracy for such classes would mean negligence of these classes which would adversely affect the quality of the software. Similarly, if we had assumed that the prediction accuracy for not change prone classes is low, it would mean that a number of not change prone classes are allocated ample resources but these resources are wasted as they are not required for managing such classes. This would lead to overshooting of schedule deadlines and budget overruns leading to poor reputation for the software organization. Thus, it is crucial for software practitioners to effectively ILP where both types of misclassifications are given equal importance so that good quality products are delivered within the allocated time and budget.

4 Experimental Design

This section states and explains the various design decisions of the study.

4.1 Variables Selection

In order to develop change prediction model, the study uses 18 metrics which represent various OO characteristics. These metrics are used to predict a software

quality attribute i.e. change proneness. The sections below briefly describe the independent and dependent variables of the study.

4.1.1 Independent Variables: OO Metrics

OO metrics are important to characterize various software properties like its cohesiveness, size, cohesion, reusability etc. These metrics are used to understand the product and manage it. The various OO metrics used in the study are briefly described below.

- Chidamber and Kemerer (CK) metrics suite: This suite comprises of six popularly used metrics namely Response For a Class (RFC), Depth of Inheritance Tree (DIT), Number of Children (NOC), Weighted Methods of a Class (WMC), Lack of Cohesion in Methods (LCOM) and Coupling Between Objects (CBO) (Chidamber and Kemerer 1994). It is a widely used metric suite used by a number of studies for developing effective software quality prediction models (Elish and Al-Khiaty 2013; Giger et al. 2012; Singh et al. 2009).
- Quality Model for Object-Oriented Design (QMOOD) metrics suite: Certain metrics from this metrics suite namely Number of Public Methods (NPM), Method of Functional Abstraction (MFA), Measure of Aggression (MOA), Cohesion Among Methods of a class (CAM) and Data Access Metric (DAM) are used in the study for developing an efficient software change prediction model. There have been certain previous studies in literature which have already used this metrics suite for quality prediction tasks (Eski and Buzluca (2011); Olague et al. (2007)).
- Other Metrics: The study also evaluates a set of other miscellaneous metrics which include Afferent Coupling (Ca) and Efferent Coupling (Ce) metrics proposed by Martin (2002). Some other commonly used metrics in literature such as Lines of Code (LOC), Coupling Between Methods of a Class (CBM), Average Method Complexity (AMC), LCOM3 (proposed by Henderson-Sellers 1996) and Inheritance Coupling (IC) are also used in the study for developing change prediction models.

In order to refer to detailed definition of the metrics used in the study, refer to http://gromit.iiar.pwr.wroc.pl/p_inf/ckjm/metric.html.

4.1.2 Dependent Variable

This study analyzes a binary dependent variable i.e. change proneness, which symbolizes the change prone nature of a class. In order to comprehend the change prone nature of a class, we analyze two different versions of the same software, say an old version and a more recent new version. A class is termed as change prone if certain LOC have been added, deleted or modified in the corresponding class, in the new version of the software as compared to the old version, otherwise it is termed as not change prone. Previous studies have also characterized the change proneness dependent variable in the same way (Lu et al. 2012; Malhotra and Khanna 2013) as taking into account the change in LOC is a practical measure for determining the binary variable change proneness. The objective of the study is to build a prediction model that predicts the outcome label for unforeseen classes.

4.2 Subject Selection

The study uses three application packages of the Android data set and three common software developed by Apache. Android is an extremely popular operating system for mobile devices, which is open source in nature and is developed by Google. This study uses three application packages of Android system namely Calendar, Bluetooth and MMS which are used for application specific functions. The data collected by Android systems is analyzed from two different versions of Android i.e. Ice Cream Sandwich and Jelly Beans. Furthermore, the study also evaluates three apache data sets namely IO, Net and Log4j. Apache Net implements a number of internet protocols on the client side. Apache IO implements a number of utilities which help in input-output functionality. These utilities may include input, output, comparator, file monitor etc. Apache Log4j software is used to embed logging statements at runtime for debugging purposes. Both Android and Apache uses GIT as the version control system. Moreover, the open source nature of the selected data sets aids the replicability of the study. Wide use of Android and apache data sets enhances the external validity of the study as the results of the study can be applied effectively in similar scenarios.

4.3 Statistical Test Selection

In order to assess and validate various hypothesis of the study, we use Friedman statistical test which is followed by Wilcoxon post-hoc test. These tests are non-parametric and thus can be effectively used without much inconvenience as they do not depend on a number of assumptions of underlying data such as data normality, homogeneity of variances etc., which are mandatory for parametric tests (Demšar 2006). Moreover, as reported by Lessmann et al. (2008), very few studies evaluate the results statistically while comparing various developed models. Thus, statistical analysis is important to strengthen the conclusions of the study.

The Friedman test is used to ascertain whether there is any significant differences in the change prediction models developed using various sampling methods (RQ1). Furthermore, the Friedman test assigns a mean rank to different sampling methods depending on their performance metrics. If the results of Friedman test are significant, we should perform post-hoc Wilcoxon test to ascertain pairwise differences amongst different sampling methods. The Wilcoxon test calculates the differences amongst the performance of different pairs of sampling methods for comparison. These computed differences are then allocated ranks in accordance with their absolute values. This study also uses Wilcoxon test for evaluating the best sampling method with MetaCost learners (RQ3).

4.4 Data Collection

This study uses six data sets which are collected using Defect Collection and Reporting System (DCRS) tool. The tool was developed in Java language by undergraduate students of Delhi Technological University. The tool is used to collect data from various open source repositories with the precondition that the repository uses GIT as its versioning control system (Malhotra et al. 2014). In order to analyze the source code of a particular data, the DCRS tool extracts change-logs between two consecutive versions of the software data set. These change logs consist of change records which specify change information including change description, its identifier, file listing of the modified files, commit timestamp, listing of changed lines of code etc.

Each data set consists of a number of data points, where each data point corresponds to OO metrics of a class and its change statistics. The change statistics include the number of insertions, deletions, modifications and the total change in a specific class of a software when two versions of a software are compared. It also has a binary variable which depicts the change prone nature of a class. This variable named ALTER is "yes" if there is a change in certain LOC of a class and "no" otherwise. In order to compute the values of different OO metrics, the DCRS tool uses another open source tool CKJM (http://gromit.iiar.pwr.wroc.pl/p_inf/ckjm/metric.html).

This study uses three application packages of Android OS and three Apache software, which use Git as their versioning control system. In order to analyze change, we use only Java class files, ignoring all other files related to media, layout etc. Table 1 lists the details of each software data set which includes the two versions analyzed, number of data points in each data set and the percentage of change prone classes. The data points correspond to the number of common classes in the two analyzed versions of each data set. A number of studies in literature have analyzed imbalanced data of varying nature i.e. data sets which are majorly imbalanced, moderately imbalanced to data sets which are mildly imbalanced (Hulse et al. 2009; Lopez et al. 2013; Liu et al. 2006). This study uses data sets in which percentage of change prone classes is in the range of 6–37%, indicating the use of highly imbalanced to moderately imbalanced data sets for developing change prediction models.

4.5 Performance Measures Selection

In order to evaluate models developed using imbalanced data, we need appropriate performance metrics. Previous studies in literature have criticized the use of traditional performance evaluators like accuracy and precision when evaluating models using imbalanced data sets (He and Garcia 2009; Gao et al. 2015; Menzies et al. 2007b). However, a number of studies in literature have discussed the favorability of using other robust performance metrics for assessing models developed using imbalanced data. Studies by Harman et al. (2014), Kubat and Matwin (1997) and He and Garcia (2009) support the use of G-mean as a performance metric for imbalanced data sets. Certain other studies by Menzies et al. (2007a) and Li et al. (2012) advocate balance performance metric as an efficient evaluator. Moreover, studies by Lessmann et al. (2008), Shatnawi (2012) and He and Garcia (2009) propose the use of ROC analysis, where Area Under ROC Curve (AUC) can be used as an effective indicator for evaluating models developed using imbalanced data. AUC is robust in handling skewness in class distributions and the unequal cost of misclassification errors (Fawcett 2006). Apart from

Table 1 Software data set details

Software name	Versions analyzed	No. of data points	% of changed classes
Android calendar	4.3.1–4.4.2	106	19%
Android bluetooth	4.3.1–4.4.2	72	19%
Android MMS	2.3.7–4.0.2	195	30%
Apache IO	2.3–2.4	198	6%
Apache net	3.0–3.1	240	37%
Apache Log4j	1.2.16–1.2.17	350	25%

the above mentioned performance metrics, the study also evaluates traditional performance metrics namely accuracy, recall and precision.

The various performance metrics used in the study can be evaluated using confusion matrix which is stated as follows. A confusion matrix for a problem with two classes namely change prone and not change prone is shown in Table 2. The positives are considered as change prone classes and the negatives are not change prone classes.

Table 3 describes the various performance metrics assessed in the study along with the formula for computing them.

5 Experimental Design

This section describes the experimental design used in the study. Figure 1 presents the pictorial representation of the experimental design. The experiment is conducted in three phases which are described as follows.

5.1 Data Pre-processing and Feature Selection

This phase involves three steps. The first step focuses on analysing the descriptive statistics of different OO metrics for each data set. This helps in assessing data characteristics of each data set. The next step focuses on identification of outliers from each data set. Outlier identification is critical in order to develop effective models which are not influenced by extreme variability in data points. This study removed outliers using the Inter Quartile range filter, which computes outliers by assessing the difference between upper quartile (Q_3) and lower quartile (Q_1). A specific data point is termed as an outlier if any of the OO metric i.e. independent variables (mentioned in section 4.1.1) in the data point is considered as an outlier. Outliers from each of the six data sets are identified and removed in order to provide effective data for model development.

The final step of this phase involves use of Correlation based Feature Selection (CFS) method in order to eliminate noisy and redundant independent variables from each data set. According to Hall (2000), the CFS method helps in identifying those independent variables which are highly correlated with the change in the class but not amongst each other. The set of independent variables extracted after application of the CFS method are important as they help in reducing the dimensionality of data sets. The CFS method evaluates an exhaustive possible combination of OO metrics, to select the best subset of OO metrics for change prediction in each corresponding data set. It may be noted that there is no single best technique for selecting features in a data set (Gao et al. 2015; Hall and Holmes 2003). A study by Hall and Holmes (2003), evaluated six attribute selection methods on 15 data sets and confirmed that though there is no best method for attribute selection, the CFS method shows good results. Furthermore, an extensive review conducted by Malhotra (2015) of 64 defect prediction

Table 2 Confusion matrix

	Predicted positive	Predicted negative
Actual positive (Change Prone)	True Positive (TP)	False Negative (FN)
Actual negative (Not Change Prone)	False Positive (FP)	True Negative (TN)

Table 3 Performance metrics

Performance metric	Description	Formula
Accuracy	It is defined as the percentage of classes which are correctly predicted. These classes include both positive and negative classes.	$\frac{TP+TN}{TP+FP+FN+TN}$ *100
Recall	It is defined as the percentage of correctly predicted positives.	$\frac{TP}{TP+FN}$ *100
Specificity	It is defined as the percentage of correctly predicted negatives.	$\frac{TN}{FP+TN}$ *100
Precision	It is defined as the percentage of correctly predicted positives amongst total number of predicted positives.	$\frac{TP}{TP+FP}$ *100
G-mean	It is the geometric mean of both positive as well as negative accuracy.	$\sqrt{\frac{TP}{TP+FP} * \frac{TN}{TN+FN}}$
Balance	It depicts the Euclidean distance between a pair of (Recall, PF), where PF corresponds to Probability of False Alarm to that of an optimal value of Recall =1 and PF = 0. PF corresponds to the percentage of incorrectly predicted change prone classes amongst actual not change prone classes.	$1-\sqrt{\frac{(0-(PF/100))^2+(1-(PD/100))^2}{2}}$ *where* $PF=\frac{FP}{FP+TN}$ *100
Area under ROC curve (AUC)	ROC represents a plot between 1-specificity values on the x-axis and recall values on the y-axis. A model is considered favorable if it has higher area under the ROC curve.	

studies from 1991 to 2013 points out that CFS was the most commonly used feature selection in those studies. Since a large number of studies in literature have successfully applied the CFS method for feature selection (Arisholm et al. 2010; Carvalho et al. 2010; Malhotra and Khanna 2013) while developing predictive models using ML techniques, we use this method.

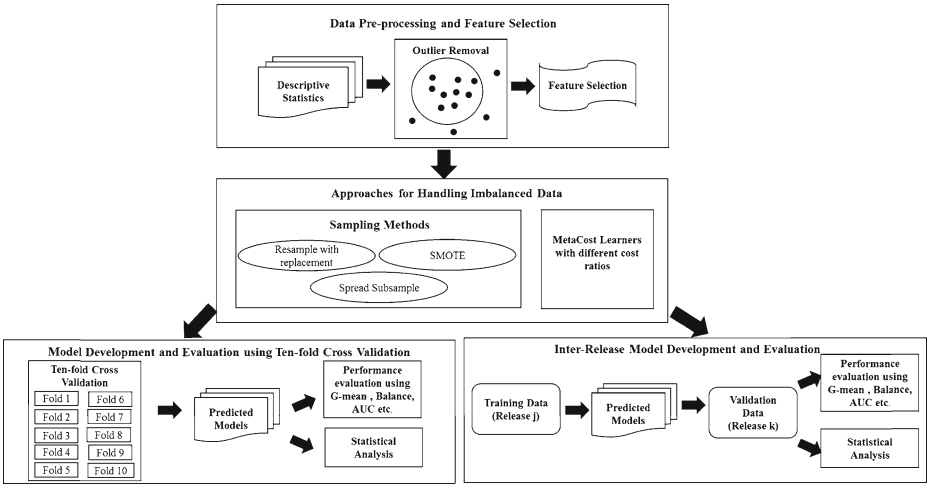

Fig. 1 Experimental design

5.2 Approaches for Handling Imbalanced Data

This phase includes application of various methods in order to treat imbalanced data and provide efficient and balanced data for training using sampling methods or provide cost-sensitization to various misclassification errors so that the model minimizes the various errors of misclassification. Thus, this phase involves use of sampling methods or MetaCost learners for efficiently handling imbalanced data and developing effective software quality models. The study evaluates three sampling methods and MetaCost learners.

5.3 Model Development and Evaluation

This phase involves development of change prediction models using six ML techniques on each of the six data sets of the study. In order to develop change prediction models, we use ten-fold cross validation technique (Stone 1974). This technique functions by performing division of all the data points of a specific data set into ten subsets. The technique then performs ten iterations by using nine subsets for training and the tenth subset for validation until we use each of the ten subsets once for validation purposes. We use ten-fold cross validation technique as it reduces validation bias (Carvalho et al. 2010; Pai and Dugan 2007). Apart from using ten-fold cross validation technique, the study also uses inter-release model validation for developing change prediction models. For developing models using inter-release validation, we train the model using a specific version of the data set and validate the model on another version of the data set. It should be noted that outliers are only removed from the training data sets while developing inter-release validation models. The next step in this phase is to evaluate the performance of the developed change prediction models. For this purpose, we use three effective performance evaluators i.e. G-mean, balance and AUC as discussed in Section 4.5. Furthermore, these performance metrics are compared with three traditional performance metrics i.e. accuracy, recall and precision. In order to increase the conclusion validity of the study, we also perform statistical testing to compare the performance of different approaches for handling imbalanced data.

6 Research Methodology

The study uses three different sampling methods along with MetaCost learners for handling ILP. The change prediction models on each data set are developed using six ML techniques. The ML techniques used in the study include Multilayer Perceptron (MLP), Random Forests (RF), Naïve Bayes (NB), Adaboost (AB), Logitboost (LB) and Bagging (BG). The study uses the default parameter settings of WEKA tool (http://www.cs.waikato.ac.nz/ml/weka/) for simulation.

6.1 Machine Learning Techniques

This section states a brief description of the various ML techniques used in the study along with the parameters used for each ML technique.

MLP are artificial neural networks which map a set of inputs from the input layer to the desired set of outputs using a number of hidden layers (Haykin 2004). They are feedforward networks which simulate the biological neurons. Backpropagation is the most common mechanism to train MLP where two passes take place i.e. forward and backward through the network.

In the forward pass, the inputs are applied and the output is produced, which is the actual response of the network. During the process, the synaptic weights of the network are adjusted. The backward pass propagates the error signal through the network which is the difference of actual output and the desired output. The weights of the network are again readjusted so that the actual response becomes closer to the desired response. The default parameter settings for MLP used in WEKA are learning rate of 0.3, momentum of 0.2 and a validation threshold of 20.

RF, proposed by Brieman (2001) is a collection of many decision trees. It is an ensemble classifier which outputs the class which is the mode of the output of all individual decision trees. This property helps in overcoming the overfitting characteristic of decision trees. The default parameter settings for RF include 100 decision trees in the random forest.

NB is a probabilistic classifier. It is based on the assumption that various input features are independent and uses Bayes theorem. The classification models developed using NB techniques are easily interpretable and are highly useful when there are large number of input features (Murphy 2006). The default settings of WEKA do not use kernel estimator or supervised discretization.

AB is a boosting method which improves the efficiency of poor classifiers and transforms them into strong classifiers. The algorithm provides a weighted sum of various weak learners to construct an output which is of an efficient boosted classifier. The initial step involves providing equal weights to all the classifiers (Witten et al. 2011). However, with each iteration, the weights are adjusted in such a manner that the algorithm focuses on the hard to learn examples to improve the efficiency of the boosted classifier. The default parameter settings for WEKA tool includes Decision Stump classifier, 10 iterations and a weight threshold of 100.

LB is another boosting method which improves the efficiency of weak learners to construct an efficient classifier. The process involves reweighting of training instances iteratively to produce a strong classifier. The LB technique uses AB technique for additive model and applies the cost functional of the logistic regression technique (Friedman et al. 2000). The default parameter settings for LB technique in WEKA include the Decision Stump classifier, 10 iterations, weight threshold of 100 and a likelihood threshold of −1.79.

BG also known as bootstrap aggregating is a technique which constantly improves the developed classification models by creating a number of versions of training set. Breiman (1996) proposed creating bootstrap duplicates of the training set, where sampling is done with replacement and a new function is trained for each of the training set. In order to predict a class, the result of majority is output. The default settings for BG technique in WEKA include REP tree as the classifier, a bag size percent of 100 and 10 iterations.

6.2 Data Sampling Methods

This section briefly describes the three sampling methods used in the study.

6.2.1 Resample with Replacement Method

The resample method is based on the bootstrapping approach. This method creates a number of subsamples, which are random in nature by using with replacement method. This is done in order to influence the original class distribution and achieve a more uniform class distribution. In with replacement method, the minority class is oversampled in order to achieve a uniform number of classes of both change prone and not change prone nature. Since, each time a different sample is

created in both the methods which is random in nature, we perform ten iterations for these methods and report the average results. The method biases the data set to produce uniform ratio of both change prone and not change prone instances.

6.2.2 Spread Subsample

This method is used to adjust the spread of the two classes by removing instances of the class which is in majority. This would help in balancing the corresponding ratios of both the classes (Tan et al. 2015). This study evaluates a number of ratios and selects the best ratio which may vary from 1:1–10:1.

6.2.3 SMOTE

SMOTE is an oversampling method where the minority classes are oversampled to increase the number of instances which belong to the minority class. The SMOTE (Synthetic Minority Oversampling Technique) method is used to create artificial instances by oversampling (Chawla et al. 2002). The method selects k-nearest neighbours randomly for oversampling the minority class. This study uses the value of $k = 5$. The next step involves selection of neighbours which is dependent on the amount of oversampling. For example, 200% oversampling would mean, two out of five neighbours are selected. Then, a synthetic sample is forged in the direction of each chosen neighbour. The creation of synthetic sample involves two steps. The first step computes the difference between chosen sample and the selected nearest neighbour. The second step multiplies the computed difference with a random number between 0 and 1, which is then added to the original sample. Such generated samples generalize the decision region of the minority class by creation of new artificial instances (Chawla et al. 2002). This study analyses five different percentage oversampling of the minority classes i.e. we evaluate SMOTE 100, SMOTE 200, SMOTE 300, SMOTE 400 and SMOTE 500.

6.3 MetaCost Learners

As discussed in section 3, both types of misclassification errors i.e. the error which occurs because of incorrect prediction of change prone class as non-change prone or the error which occurs because of misclassifying a non-change prone class as change prone should be minimized and balanced. A MetaCost learner is used to cost-sensitize a specific classification technique by wrapping it up in another procedure which is used to minimize cost. The cost of misclassifying a class ‘a’ as belonging to class ‘b’ is denoted by C(a,b). The MetaCost learners use Bayes optimal prediction in order to minimize the conditional risk (R(a|x)). The risk ascertains the expected cost of classifying class x as belonging to category ‘a’ as shown in Eq. (1). The P(a|x) in Eq. (1) is the probability that a specific class x belongs to class category ‘a’. It is certain that Bayes optimal prediction would lead to lowest possible cost for all the instances x. The space of all examples ‘X’ is partitioned into ‘n’ regions. The regions are such that class ‘a’ is the most optimal prediction in terms of cost in region ‘a’.

$$R\left(a\middle|x\right) = \sum_{n} P\left(a\middle|x\right) C(a,b) \tag{1}$$

The MetaCost learners are such that they modify the labels of training examples so that they are representatives of their optimal class. This is done by using an ensemble of ‘k’ classifiers

for learning and using the vote of each classifier. It is basically a variation of bagging technique with replacement. The labels are allocated to each example on the basis of votes received by it or by taking into account the probability estimates of 'k' models ($k = 10$). Therefore, the allocation of a new class label is dependent on the cost ratio and the probability estimates. The new training set with relabelled examples is then used by the base classification technique to develop a prediction model which is cost sensitive (Seliya and Khoshgoftaar 2011; Domingos 1999).

7 Analysis and Results

This section states the analysis steps performed to evaluate the change prediction models. It describes the results of the study.

7.1 Descriptive Statistics

After analysing the descriptive statistics of all the 18 OO metrics for each of the six data sets, which include the minimum, maximum, mean, standard deviation, 25% percentile and 75% percentile, we summarize the following observations:

- The mean values of NOC (Calendar-0.95, Bluetooth-1, MMS-0.85, Apache IO-0.28, Net-0 and Log4j-0.20) and DIT (Calendar-0.37, Bluetooth-0, MMS-0.19, Apache IO-0.73, Net- 1 and Log4j-0.95) suggest that reusability through inheritance was not much used in the data sets. Cartwright and Shepperd (2000) point out a similar trend in their study.
- Each data set had varying class size in terms of LOC. The class size for Calendar data set varied between 8 and 1507 LOC, for Bluetooth data set 9–718 LOC and 0–882 for MMS data set. For Apache data sets the class sizes were in the range 8–768 LOC for IO data set, 0–2372 for Net data set and 1–1864 for Log4j data set.
- An analysis of LCOM values indicated that the data sets exhibit significantly high LCOM values with maximum LCOM values as 5995 for Calendar, 6903 for Bluetooth, 8128 for MMS, 8001 for Apache IO, 6252 for Net and 7503 for Log4j. The mean values of CAM metric varied in the range 0.37–0.51. The values of MFA metric were very low indicating hardly any use of functional abstraction.

7.2 Outlier Analysis

In order to develop effective change prediction models, it is important to identify and remove outliers in the data. Thus, to provide impartial results, we find all outliers using Inter Quartile range filter of WEKA tool and remove them from each data set. The number of outliers determined in each data set are shown in Table 4.

7.3 Correlation Based Feature Selection Results

The independent variables selected by each data set, after the application of CFS method are shown in Table 5 below. According to the table, the most commonly selected OO metric for predicting change prone classes was LOC. Other commonly used metrics were AMC and

Table 4 Outlier details

Data set	Number of outliers
Android calendar	6
Android bluetooth	7
Android MMS	22
Apache IO	11
Apache net	39
Apache Log4j	65

LCOM3, which were selected in three out of six data sets of the study. RFC, WMC, MOA and CAM were selected by two data sets each, indicating their ability to determine change prone nature of a class.

7.4 Discussion and Evaluation of Ten-Fold Cross Validation Results

RQ1a: Does the performance of different ML methods in this study significantly improve by using various sampling approaches for ILP in change prediction?
RQ1b: Which sampling approach performs best amongst different sampling techniques analysed in the study?

In order to assess the change prediction models developed using various sampling methods we use accuracy, recall, precision, G-mean, balance and AUC performance metrics.

Tables 6, 7, 8, 9, 10, and 11 report the values of different performance metrics as shown by the six ML techniques on each of the data set for each sampling method and the scenario where no sampling method is used. It should be noted that SMOTE and spread subsample values are those which are best for a particular data set. Since, a number of combinations of SMOTE and spread subsample are explored, we choose and compare those which give the best results for a corresponding data set.

According to Table 6, the accuracy values of the scenario where no sampling method is used is better than the one in which we use certain sampling methods for handling imbalanced learning in two data sets namely Calendar and IO, and for few cases in Log4j data set. However, accuracy is not an effective indicator of performance evaluation, specifically in the case of imbalanced data sets. This is discussed in a number of literature studies (He and Garcia 2009; Gao et al. 2015; Weng and Poon 2008). However, in all other data sets, it can be

Table 5 CFS results

Data set	Independent variables selected
Android Calendar	CBO, Ce
Android Bluetooth	WMC, RFC, LOC, CAM
Android MMS	LCOM3, LOC, DAM, MOA, CAM, AMC
Apache IO	NPM, LCOM3, AMC
Apache Net	WMC, NOC, LCOM3, LOC
Apache Log4j	RFC, LOC, MOA, AMC

Table 6 Accuracy results using different sampling methods

ML technique	Calendar				ML technique	Bluetooth			
	Resample	SMOTE	Subsample	No Sample		Resample	SMOTE	Subsample	No Sample
MLP	71.03	51.89	82.56	85.00	MLP	87.09	81.65	73.91	83.08
RF	80.17	76.76	79.07	83.00	RF	96.62	82.57	82.61	78.46
NB	54.62	55.14	79.07	81.00	NB	83.85	78.90	86.96	83.08
AB	74.45	56.22	82.56	85.00	AB	95.23	78.90	86.96	75.38
LB	77.86	74.59	82.56	85.00	LB	96.61	86.24	82.61	78.46
BG	78.04	69.73	81.40	83.00	BG	91.08	80.73	82.61	83.08
ML technique	MMS				ML technique	IO			
	Resample	SMOTE	Subsample	No Sample		Resample	SMOTE	Subsample	No Sample
MLP	88.43	86.03	75.17	75.72	MLP	81.55	84.85	88.89	95.72
RF	95.72	89.95	69.66	71.68	RF	98.18	85.71	86.67	91.44
NB	77.86	78.19	69.66	70.52	NB	74.65	78.79	84.44	93.58
AB	79.59	85.05	73.10	69.94	AB	81.44	86.58	82.22	93.05
LB	84.34	85.05	73.10	73.99	LB	88.93	85.71	82.22	94.12
BG	89.88	87.75	76.55	78.61	BG	94.12	84.85	88.89	94.65
ML technique	Net				ML technique	Log4j			
	Resample	SMOTE	Subsample	No Sample		Resample	SMOTE	Subsample	No Sample
MLP	88.75	77.78	77.47	76.62	MLP	77.89	63.82	84.23	81.40
RF	94.82	85.19	75.27	74.63	RF	88.98	81.14	79.65	78.60
NB	72.45	66.40	74.18	74.13	NB	67.54	59.21	78.95	79.65
AB	78.06	81.75	76.92	77.11	AB	74.60	71.27	82.46	82.46
LB	87.36	83.60	75.27	78.11	LB	79.96	76.32	82.46	82.81
BG	88.76	84.92	76.92	77.61	BG	85.36	79.17	81.40	81.40

Table 7 Recall results using different sampling methods

ML technique	Calendar				ML technique	Bluetooth			
	Resample	SMOTE	Subsample	No Sample		Resample	SMOTE	Subsample	No Sample
MLP	56.73	51.96	17.65	17.65	MLP	81.15	89.09	72.73	27.27
RF	77.27	67.65	17.65	5.88	RF	98.28	83.64	81.82	27.27
NB	22.36	30.39	29.41	29.41	NB	66.55	70.91	81.82	36.36
AB	67.46	78.43	17.65	17.65	AB	94.48	80.00	81.82	9.09
LB	78.73	67.65	17.65	17.65	LB	97.59	85.45	81.82	27.27
BG	74.55	68.63	11.76	17.65	BG	87.93	78.18	81.82	9.09
ML technique	MMS				ML technique	IO			
	Resample	SMOTE	Subsample	No Sample		Resample	SMOTE	Subsample	No Sample
MLP	93.23	93.62	51.06	44.68	MLP	94.73	38.18	54.55	27.27
RF	97.21	95.74	44.68	40.43	RF	99.89	60.00	63.64	27.27
NB	86.51	83.69	72.34	70.21	NB	71.54	29.09	45.45	27.27
AB	90.69	96.10	68.09	51.06	AB	88.79	60.00	45.45	27.27
LB	88.60	95.39	61.70	38.30	LB	94.62	58.18	54.55	27.27
BG	92.21	96.10	55.32	40.43	BG	97.58	49.09	63.64	9.09
ML technique	Net				ML technique	Log4j			
	Resample	SMOTE	Subsample	No Sample		Resample	SMOTE	Subsample	No Sample
MLP	89.05	88.56	45.76	40.68	MLP	64.41	53.51	22.81	17.54
RF	94.10	90.68	62.71	50.85	RF	87.21	74.12	31.58	26.32
NB	56.42	54.66	38.98	32.20	NB	37.79	30.26	10.53	14.04
AB	80.53	94.07	61.02	61.02	AB	61.91	59.21	22.81	24.56
LB	87.68	91.53	54.24	57.63	LB	71.91	71.49	22.81	26.32
BG	87.89	91.53	55.93	55.93	BG	81.91	75.00	17.54	19.30

Table 8 Precision results using different sampling methods

ML technique	Calendar				ML technique	Bluetooth			
	Resample	SMOTE	Subsample	No Sample		Resample	SMOTE	Subsample	No Sample
MLP	76.39	56.99	75.00	75.00	MLP	89.56	77.78	72.73	50.00
RF	80.16	87.34	42.86	50.00	RF	94.59	82.14	81.82	33.33
NB	52.70	72.09	45.45	41.67	NB	95.94	84.78	90.00	50.00
AB	76.38	57.55	75.00	75.00	AB	94.83	78.57	90.00	14.29
LB	76.22	83.13	75.00	75.00	LB	95.04	87.04	81.82	33.33
BG	78.21	74.47	66.67	50.00	BG	92.03	82.69	81.82	50.00
ML technique	MMS				ML technique	IO			
	Resample	SMOTE	Subsample	No Sample		Resample	SMOTE	Subsample	No Sample
MLP	85.37	87.13	64.86	56.76	MLP	74.50	95.45	100	100.00
RF	95.79	90.30	53.85	47.50	RF	96.53	75.00	77.78	27.27
NB	79.54	84.59	52.31	47.14	NB	76.35	61.54	83.33	42.86
AB	81.57	84.42	57.14	45.28	AB	76.97	78.57	71.43	37.50
LB	84.98	84.86	58.00	52.94	LB	84.82	76.19	66.67	50.00
BG	90.13	87.42	66.67	67.86	BG	91.02	79.41	87.50	100.00
ML technique	Net				ML technique	Log4j			
	Resample	SMOTE	Subsample	No Sample		Resample	SMOTE	Subsample	No Sample
MLP	87.46	78.57	75.00	66.67	MLP	86.11	67.4	100.00	62.50
RF	95.03	86.29	61.67	57.69	RF	89.58	86.22	48.65	44.12
NB	79.76	86.58	67.65	61.29	NB	87.23	71.88	40.00	47.06
AB	75.63	80.14	65.45	61.02	AB	81.50	78.03	68.42	66.67
LB	85.95	83.72	64.00	64.15	LB	84.02	79.13	68.42	68.18
BG	88.46	85.38	67.35	63.46	BG	86.72	81.82	62.50	61.11

Table 9 G-mean results using different sampling methods

ML technique	Calendar				ML technique	Bluetooth			
	Resample	SMOTE	Subsample	No Sample		Resample	SMOTE	Subsample	No Sample
MLP	0.72	0.52	0.78	0.80	MLP	0.88	0.82	0.74	0.66
RF	0.80	0.78	0.59	0.65	RF	0.97	0.83	0.83	0.53
NB	0.54	0.60	0.62	0.59	NB	0.87	0.79	0.87	0.66
AB	0.75	0.55	0.79	0.80	AB	0.95	0.79	0.87	0.34
LB	0.79	0.75	0.79	0.80	LB	0.96	0.86	0.82	0.53
BG	0.78	0.69	0.74	0.65	BG	0.91	0.81	0.82	0.65
ML technique	MMS				ML technique	IO			
	Resample	SMOTE	Subsample	No Sample		Resample	SMOTE	Subsample	No Sample
MLP	0.89	0.85	0.71	0.68	MLP	0.83	0.89	0.93	0.98
RF	0.96	0.90	0.64	0.61	RF	0.98	0.81	0.83	0.51
NB	0.79	0.74	0.66	0.64	NB	0.76	0.71	0.84	0.64
AB	0.81	0.86	0.68	0.61	AB	0.82	0.83	0.78	0.59
LB	0.85	0.85	0.68	0.65	LB	0.89	0.82	0.76	0.69
BG	0.90	0.88	0.73	0.74	BG	0.94	0.83	0.88	0.97
ML technique	Net				ML technique	Log4j			
	Resample	SMOTE	Subsample	No Sample		Resample	SMOTE	Subsample	No Sample
MLP	0.89	0.77	0.77	0.72	MLP	0.79	0.64	0.91	0.72
RF	0.95	0.85	0.71	0.68	RF	0.89	0.82	0.64	0.61
NB	0.74	0.68	0.72	0.68	NB	0.74	0.63	0.57	0.62
AB	0.79	0.83	0.73	0.72	AB	0.76	0.72	0.76	0.75
LB	0.87	0.84	0.71	0.73	LB	0.81	0.77	0.75	0.76
BG	0.89	0.85	0.74	0.72	BG	0.85	0.79	0.72	0.71

Table 10 Balance results using different sampling methods

ML technique	Calendar				ML technique	Bluetooth			
	Resample	SMOTE	Subsample	No Sample		Resample	SMOTE	Subsample	No Sample
MLP	66.55	51.88	41.76	41.76	MLP	85.16	80.11	73.84	48.42
RF	79.24	75.59	41.62	33.44	RF	96.37	82.53	82.56	47.98
NB	43.34	49.73	49.71	49.73	NB	76.23	77.48	85.86	54.70
AB	71.32	47.47	41.76	41.76	AB	94.85	78.86	85.86	35.24
LB	75.67	74.20	41.76	41.76	LB	96.49	86.22	82.56	47.98
BG	76.28	69.83	37.60	41.71	BG	89.95	80.59	82.56	35.70
ML technique	MMS				ML technique	IO			
	Resample	SMOTE	Subsample	No Sample		Resample	SMOTE	Subsample	No Sample
MLP	87.43	77.65	64.15	59.87	MLP	77.62	56.29	67.86	48.57
RF	95.32	83.45	58.78	56.26	RF	97.56	71.37	73.95	48.47
NB	76.23	73.25	70.29	70.42	NB	70.52	49.70	61.37	48.55
AB	76.83	71.80	71.55	61.76	AB	79.49	71.49	61.21	48.53
LB	83.62	72.87	68.97	55.46	LB	86.96	70.16	67.26	48.56
BG	89.42	77.94	67.04	57.57	BG	93.16	63.89	74.20	35.72
ML technique	Net				ML technique	Log4j			
	Resample	SMOTE	Subsample	No Sample		Resample	SMOTE	Subsample	No Sample
MLP	88.54	70.49	61.30	57.63	MLP	73.45	45.42	62.38	41.67
RF	94.45	81.83	70.50	63.56	RF	88.65	51.26	79.88	47.57
NB	67.67	66.43	56.39	51.69	NB	55.80	36.67	49.98	39.15
AB	77.10	72.29	70.35	70.15	AB	70.82	45.38	68.84	46.61
LB	87.11	78.24	66.03	68.58	LB	77.85	45.38	75.83	47.85
BG	88.47	80.63	67.51	67.43	BG	84.61	41.67	78.75	42.89

Table 11 AUC results using different sampling methods

ML technique	Calendar				ML technique	Bluetooth			
	Resample	SMOTE	Subsample	No Sample		Resample	SMOTE	Subsample	No Sample
MLP	0.75	0.56	0.51	0.53	MLP	0.87	0.88	0.77	0.79
RF	0.87	0.82	0.55	0.61	RF	0.97	0.88	0.89	0.70
NB	0.59	0.56	0.56	0.58	NB	0.91	0.90	0.95	0.85
AB	0.80	0.61	0.46	0.43	AB	0.96	0.83	0.83	0.66
LB	0.84	0.76	0.47	0.49	LB	0.96	0.91	0.83	0.67
BG	0.84	0.79	0.54	0.54	BG	0.95	0.88	0.89	0.77
ML technique	MMS				ML technique	IO			
	Resample	SMOTE	Subsample	No Sample		Resample	SMOTE	Subsample	No Sample
MLP	0.90	0.88	0.79	0.80	MLP	0.88	0.85	0.84	0.74
RF	0.98	0.92	0.780	0.61	RF	1.00	0.92	0.81	0.67
NB	0.82	0.82	0.79	0.79	NB	0.88	0.82	0.85	0.72
AB	0.87	0.89	0.78	0.76	AB	0.93	0.87	0.84	0.66
LB	0.91	0.89	0.82	0.79	LB	0.98	0.88	0.86	0.70
BG	0.96	0.90	0.82	0.79	BG	0.99	0.89	0.74	0.67
ML technique	Net				ML technique	Log4j			
	Resample	SMOTE	Subsample	No Sample		Resample	SMOTE	Subsample	No Sample
MLP	0.89	0.85	0.84	0.77	MLP	0.80	0.69	0.67	0.67
RF	0.98	0.92	0.83	0.80	RF	0.96	0.88	0.62	0.61
NB	0.81	0.81	0.78	0.78	NB	0.68	0.62	0.58	0.62
AB	0.84	0.87	0.80	0.78	AB	0.79	0.78	0.64	0.66
LB	0.93	0.87	0.81	0.79	LB	0.87	0.82	0.65	0.68
BG	0.95	0.90	0.81	0.78	BG	0.93	0.85	0.65	0.68

seen that in majority of the cases, the performance of change prediction models developed using sampling methods is better in terms of accuracy metric as compared to the scenario where no sampling method is used.

An analysis of Table 7 indicates extremely low recall values (5–45%) in majority of the cases on all the data sets when no sampling method is used. A similar trend was shown by precision values as indicated in Table 8. The table shows low precision values (14–65%) in majority of cases when no sampling approach was used. These trends are a result of imbalanced data. As the data sets have very few instances of change prone classes, the models may not be able to learn efficiently the prediction of change prone classes, this would result in low recall and precision values. However, in majority of cases when any sampling approach was used, the recall values ranged from 55 to 98% (Table 7), indicating a positive improvement in the correct prediction of change prone classes. Similarly, the precision values too showed a positive improvement as they ranged from 70 to 96% after applying a sampling approach in majority of cases (Table 8).

The G-mean, balance and AUC performance metrics also showed positive improvement in most of the cases when a sampling method was used as compared to the scenario when no sampling method was used. The G-mean and AUC performance metrics showed a positive improvement of up to 35% with the use of sampling methods and the balance metric of up to 45%.

In order to evaluate whether the use of different sampling methods improve the performance of various ML techniques for ILP we use Friedman test to ascertain the hypothesis given in the following sections. The lower the rank achieved by a given sampling method, the better it is considered. Since this section uses Friedman test to ascertain significant differences amongst four sampling methods namely resample with replacement, SMOTE and spread subsample along with the scenario when no sampling method is used, the degrees of freedom for Friedman test is 3. In order to test the hypothesis, the p-value is stated. The study checks the hypothesis at the cut-off $\alpha = 0.05$. In order to apply Friedman test, we used the values of performance metrics achieved by all the six ML techniques on a specific data set using different sampling approaches and without any sampling approach. This was done for each of the six data sets and the Friedman test was evaluated individually on all the data sets. All the change prediction models were assessed on three stable performance metrics namely G-mean, balance and AUC.

7.4.1 Friedman Results for Different Sampling Methods Using G-mean Measure

This section evaluates hypothesis H0 which is stated below for each data set of the study.

> *Null Hypothesis H0*: Change prediction models developed using various ML Techniques (MLP, RF, NB, AB, LB, BG) do not show significant differences when various sampling methods (Resampling with replacement, SMOTE, Spread Subsample) are used for handling an imbalanced data set as compared to use of no sampling method when evaluated using G-mean measure.
> *Alternate Hypothesis H0a*: Change prediction models developed using various ML Techniques (MLP, RF, NB, AB, LB, BG) show significant differences when various sampling methods (Resampling with replacement, SMOTE, Spread Subsample) are used for handling an imbalanced data set as compared to use of no sampling method when evaluated using G-mean measure.

Table 12 states the Friedman ranking allocated to each sampling method and a scenario where no sampling method is used, along with mean ranks allocated by Friedman test (shown in parenthesis) and its p-value, when evaluated using G-mean performance measure on each data set of the study. As depicted in the table, in five out of six data sets, the best rank is achieved by resample method with replacement. However, the worst scenario according to G-mean results was that when no sampling method was used as in five out of six data sets, the no sample scenario achieved the worst rank. The Friedman test was significant at $\alpha = 0.05$ in four data sets (Bluetooth, MMS, Net and Log4j) as a p-value of less than 0.05 was obtained. Since, in majority of the data sets, the resample with replacement method significantly outperformed the other scenarios, we reject null hypothesis H0, which states that use of different sampling methods do not differ significantly from each other when evaluated using G-mean measure.

7.4.2 Friedman Results for Different Sampling Methods Using Balance Measure

This section evaluates Hypothesis H1, which is presented below.

> *Null Hypothesis H1*: Change prediction models developed using various ML Techniques (MLP, RF, NB, AB, LB, BG) do not show significant differences when various sampling methods (Resampling with replacement, SMOTE, Spread Subsample) are used for handling an imbalanced data set as compared to use of no sampling method when evaluated using balance measure.
> *Alternate Hypothesis H1a*: Change prediction models developed using various ML Techniques (MLP, RF, NB, AB, LB, BG) show significant differences when various sampling methods (Resampling with replacement, SMOTE, Spread Subsample) are used for handling an imbalanced data set as compared to use of no sampling method when evaluated using balance measure.

Table 12 states the Friedman ranking allocated to each sampling method and a scenario where no sampling method is used, along with mean ranks allocated by Friedman test (shown in parenthesis) and its p-value, when evaluated using balance performance measure on each data set of the study. As depicted in the table, the best rank is achieved by resample method with replacement in all the six data sets. Moreover, the Friedman results were significant at

Table 12 Friedman results using G-mean measure

Data set	Rank 1	Rank 2	Rank 3	Rank 4	p-value
Calendar	Subsample (2.17)	No Sample (2.17)	Resample (2.50)	SMOTE (3.17)	0.494
Bluetooth	Resample (1.17)	Subsample (2.33)	SMOTE (2.50)	No Sample (4.00)	0.002
MMS	Resample (1.33)	SMOTE (1.67)	Subsample (3.17)	No Sample (3.83)	0.002
IO	Resample (2.00)	Subsample (2.33)	SMOTE (2.67)	No Sample (3.00)	0.572
Net	Resample (1.17)	SMOTE (2.17)	Subsample (3.00)	No Sample (3.67)	0.006
Log4j	Resample (1.17)	SMOTE (2.67)	Subsample (2.83)	No Sample (3.33)	0.006

$\alpha = 0.05$ in all the six data sets. The worst scenario according to balance results was that when no sampling method was used as in four out of six data sets the no sample scenario achieved the worst rank. As Resample with replacement method significantly out-performed all other methods in all the six data sets, it leads to acceptance of alternate hypothesis, which means that use of different sampling methods differ significantly from each other when evaluated using balance measure (Table 13).

7.4.3 Friedman Results for Different Sampling Methods Using AUC Measure

This section tests Hypothesis H2 which is stated as follows.

> *Null Hypothesis H2*: Change prediction models developed using various ML Techniques (MLP, RF, NB, AB, LB, BG) do not show significant differences when various sampling methods (Resampling with replacement, SMOTE, Spread Subsample) are used for handling an imbalanced data set as compared to use of no sampling method when evaluated using AUC measure.
> *Alternate Hypothesis H2a*: Change prediction models developed using various ML Techniques (MLP, RF, NB, AB, LB, BG) show significant differences when various sampling methods (Resampling with replacement, SMOTE, Spread Subsample) are used for handling an imbalanced data set as compared to use of no sampling method when evaluated using AUC measure.

Table 14 states the Friedman test results when evaluated on all data sets using AUC measure. It reports the ranking allocated to each sampling method and a scenario where no sampling method is used, along with mean ranks allocated by Friedman test (shown in parenthesis) and its p-value. The Friedman results were evaluated at $\alpha = 0.05$. According to Table 14, the resample method with replacement was significantly better than all the other scenarios in all the six data sets. Thus, this leads to rejection of null hypothesis, which means that use of different sampling methods differ significantly from each other when evaluated using AUC measure. Also, in four out of six data sets, the worst rank was allocated to the scenario where no sampling method was used, which indicates that using a sampling approach for an imbalanced data improves the AUC results in majority of the data sets.

Table 13 Friedman results using balance measure

Data set	Rank 1	Rank 2	Rank 3	Rank 4	p-value
Calendar	Resample (1.50)	SMOTE (1.92)	No Sample (3.17)	Subsample (3.42)	0.018
Bluetooth	Resample (1.33)	Subsample (2.17)	SMOTE (2.50)	No Sample (4.00)	0.004
MMS	Resample (1.00)	SMOTE (2.00)	Subsample (3.17)	No Sample (3.83)	0.001
IO	Resample (1.00)	SMOTE (2.00)	Subsample (3.17)	No Sample (4.00)	0.001
Net	Resample (1.00)	SMOTE (2.00)	Subsample (3.17)	No Sample (4.00)	0.001
Log4j	Resample (1.00)	SMOTE (2.00)	No Sample (3.83)	Subsample (3.67)	0.001

Table 14 Friedman Results using AUC measure

Data Set	Rank 1	Rank 2	Rank 3	Rank 4	p-value
Calendar	Resample (1.00)	SMOTE (2.17)	No Sample (3.00)	Subsample (3.83)	0.001
Bluetooth	Resample (1.33)	SMOTE (2.33)	Subsample (2.50)	No Sample (3.83)	0.010
MMS	Resample (1.17)	SMOTE (1.83)	Subsample (3.17)	No Sample (3.83)	0.001
IO	Resample (1.33)	SMOTE (2.00)	Subsample (2.67)	No Sample (4.00)	0.003
Net	Resample (1.00)	SMOTE (1.67)	Subsample (3.00)	No Sample (4.00)	0.001
Log4j	Resample (1.00)	SMOTE (2.17)	No Sample (3.00)	Subsample (3.83)	0.001

7.4.4 Wilcoxon Results for Different Sampling Methods

According to ranks allocated by Friedman test, it was found that the resample with replacement was the best sampling method in majority of the data sets when evaluated using G-mean, balance and AUC performance measures. In order to further investigate and ascertain whether, resample with replacement method is better than all the other employed sampling methods of the study, we use post-hoc Wilcoxon test with Bonferroni correction. The use of Bonferroni correction removes familywise errors. The Wilcoxon test was performed at significance level $\alpha = 0.05$. The test computes pairwise comparisons amongst resample with replacement method and the other sampling methods using G-mean, balance and AUC performance measure. The test was evaluated on the results of all the ML techniques on all the data sets of the study using a specific performance measure. The hypothesis being tested are as follows.

> *Null Hypothesis H3*: Change prediction models developed using various ML Techniques (MLP, RF, NB, AB, LB, BG) do not show significant differences in performance metrics (G-mean, balance and AUC) when Resampling with replacement sampling method is used instead of other sampling methods (SMOTE, Spread Subsample) for handling imbalanced data sets.
> *Alternate Hypothesis H3a*: Change prediction models developed using various ML Techniques (MLP, RF, NB, AB, LB, BG) show significant differences in performance metrics (G-mean, balance and AUC) when Resampling with replacement sampling method is used instead of other sampling methods (SMOTE, Spread Subsample) for handling imbalanced data sets.

Table 15 Wilcoxon test results on sampling methods performance

Pair	G-mean	Balance	AUC
Resample with replacement vs SMOTE	S+ (0.000)	S+ (0.000)	S+ (0.000)
Resample with replacement vs Spread Subsample	S+ (0.000)	S+ (0.000)	S+ (0.000)

Table 15 shows the results of Wilcoxon test where it is seen that resample with replacement sampling method is significantly better than all the other sampling methods using different performance measures (G-mean, balance, AUC). The table states the p-value of pairwise comparisons of different sampling methods with resample with replacement method. It can be seen that all other sampling methods perform not as well as resample with replacement method and the results of Wilcoxon test are statistically significant. The symbol S+ represents that the results are significantly better according to Wilcoxon test.

RQ2 What is the effectiveness of different MetaCost learners for improving learning through imbalanced data?

In order to ascertain the effectiveness of MetaCost learners, we compare the performance of different MetaCost learners using G-mean, balance, AUC and cost with the scenario when no cost-sensitive approach is applied. In order to calculate the total cost of misclassification, we sum up the misclassification costs of FP and FN predictions as given in equation for each Cost Ratio (CR).

$$\text{Total Cost} = (\text{Cost of a FN Prediction*No. of FN}) + (\text{Cost of an FP prediction*No. of FP})$$

The study analyses seven different CR's which are 5, 10, 15, 20, 25, 30 and 50. Table 16, 17, 18, 19, 20, and 21 states the values of different performance metrics obtained on application of all the six ML techniques (MLP, RF, NB, AB, LB and BG) on each data set of the study using different CR's and the "original" technique without any CR. If a specific performance metric was undefined in a specific scenario, then the value was shown as "N.D." representing "Not Defined". The total cost of each ML method on each data set using various cost ratios is reported in Table 22.

As seen in Table 16, the average performance of all the techniques on all the data sets decreased in terms of accuracy measure when a MetaCost learner was used in comparison to the scenario when no such technique was applied i.e. original ML technique was used. This trend is a result of using a biased performance metric i.e. accuracy which does not take into account the imbalanced nature of the data. Several studies have ascertained the fact that the use of accuracy metric is not suitable as it gives highly optimistic results which are not sensitized to the class distributions in the data set (He and Garcia 2009; Gao et al. 2015; Weng and Poon 2008). The accuracy metric only focuses on the majority class i.e. the large number of non-change prone classes which are correctly predicted and ignores the low number of correctly predicted change prone classes. This fact can be ascertained from the recall values as shown in Table 17. There was a sharp increase in recall values in all the data sets with the use of MetaCost learners. This is because the MetaCost learners were used in such a way so as to penalize the classifier for false negative predictions. The lower, the number of false negative predictions, the higher would be the recall. This was done as the data sets were highly imbalanced with few change prone classes. Thus, it was important to increase recall rates to achieve a balanced classifier. However, one should be careful that apart from recall, the precision values should also be considered. As shown in Table 18, there is decrease in precision values in majority of the cases. This is a concerning factor as increase in recall should not mean that a number of non-change prone classes are incorrectly classified as change prone. One should achieve an optimum balance between recall and precision. It

Table 16 Accuracy results of MetaCost learners using ML techniques

ML tech.	Calendar								ML tech.	Bluetooth							
	Original	MC5	MC10	MC15	MC20	MC25	MC30	MC50		Original	MC5	MC10	MC15	MC20	MC25	MC30	MC50
MLP	75.00	73.00	16.00	17.00	17.00	17.00	17.00	17.00	MLP	83.08	75.38	64.62	56.92	50.77	47.69	44.62	33.85
RF	50.00	68.00	33.00	31.00	29.00	26.00	25.00	23.00	RF	78.46	81.54	64.06	63.08	60.00	58.46	56.92	55.38
NB	41.67	73.00	64.00	34.00	17.00	17.00	17.00	17.00	NB	83.08	84.62	78.46	76.92	75.38	75.38	75.38	75.38
AB	75.00	82.00	17.00	16.00	16.00	17.00	17.00	17.00	AB	75.38	76.92	69.23	67.69	66.15	64.62	61.54	55.38
LB	75.00	78.00	25.00	18.00	16.00	16.00	16.00	17.00	LB	78.46	73.85	72.31	63.08	63.08	55.38	50.77	44.62
BG	50.00	74.00	17.00	17.00	17.00	17.00	17.00	17.00	BG	83.08	76.92	66.15	61.54	46.15	26.15	18.46	16.92
ML tech.	MMS								ML tech.	IO							
	Original	MC5	MC10	MC15	MC20	MC25	MC30	MC50		Original	MC5	MC10	MC15	MC20	MC25	MC30	MC50
MLP	75.72	61.75	25.26	20.00	20.00	20.00	20.00	20.00	MLP	95.72	92.51	80.75	71.66	66.31	64.17	56.15	47.59
RF	71.68	62.11	50.88	45.96	42.81	41.40	39.30	32.98	RF	91.44	86.10	86.10	83.96	82.89	81.28	78.61	75.40
NB	70.52	64.21	31.93	23.16	22.46	22.81	20.70	21.05	NB	93.58	77.54	73.26	72.73	71.12	69.52	67.91	64.71
AB	69.94	56.49	37.54	18.95	20.00	20.00	20.00	20.00	AB	93.05	89.84	89.30	85.03	78.07	77.01	76.47	69.52
LB	73.99	65.26	36.84	22.46	20.00	20.00	20.00	20.00	LB	94.12	91.44	86.10	81.28	73.26	70.05	65.78	44.39
BG	78.61	64.91	32.28	20.00	20.00	20.00	20.00	20.00	BG	94.65	91.44	90.37	86.1	82.89	75.94	62.03	16.04
ML tech.	Net								ML tech.	Log4j							
	Original	MC5	MC10	MC15	MC20	MC25	MC30	MC50		Original	MC5	MC10	MC15	MC20	MC25	MC30	MC50
MLP	76.62	66.67	62.19	52.24	44.78	38.31	38.31	31.34	MLP	81.40	61.75	25.26	20.00	20.00	20.00	20.00	20.00
RF	74.63	68.66	63.68	61.69	59.70	58.71	58.71	60.20	RF	78.60	62.11	50.88	45.96	42.81	41.40	39.30	32.98
NB	74.13	76.12	73.63	66.67	65.17	61.19	61.19	60.70	NB	79.65	64.21	31.93	23.16	22.46	22.81	20.70	21.05
AB	77.11	64.68	56.22	54.73	53.73	52.74	52.74	52.74	AB	82.46	56.49	37.54	18.95	20.00	20.00	20.00	20.00
LB	78.11	63.18	59.70	55.72	55.22	52.24	52.24	31.34	LB	82.81	65.26	36.84	22.46	20.00	20.00	20.00	20.00
BG	77.61	63.18	57.21	55.22	52.74	39.80	39.80	29.35	BG	81.40	64.91	32.28	20.00	20.00	20.00	20.00	20.00

Table 17 Recall results of MetaCost learners using ML techniques

ML tech.	Calendar								ML tech.	Bluetooth							
	Original	MC5	MC10	MC15	MC20	MC25	MC30	MC50		Original	MC5	MC10	MC15	MC20	MC25	MC30	MC50
MLP	17.65	29.41	88.24	100.00	100.00	100.00	100.00	100.00	MLP	27.27	72.73	81.82	81.82	90.91	90.91	90.91	90.91
RF	5.88	41.18	70.59	82.35	82.35	76.47	76.47	76.47	RF	27.27	72.73	80.00	81.82	81.82	81.82	81.82	81.82
NB	29.41	29.41	47.06	70.59	100.00	100.00	100.00	100.00	NB	36.36	72.73	72.73	72.73	72.73	72.73	81.82	90.91
AB	17.65	41.18	94.12	94.12	94.12	100.00	100.00	100.00	AB	9.09	72.73	72.73	72.73	72.73	81.82	81.82	81.82
LB	17.65	41.18	88.24	94.12	94.12	94.12	94.12	100.00	LB	27.27	81.82	72.73	72.73	72.73	72.73	72.73	81.82
BG	17.65	29.41	100.00	100.00	100.00	100.00	100.00	100.00	BG	9.09	72.73	81.82	81.82	81.82	81.82	90.91	100
ML tech.	MMS								ML tech.	IO							
	Original	MC5	MC10	MC15	MC20	MC25	MC30	MC50		Original	MC5	MC10	MC15	MC20	MC25	MC30	MC50
MLP	44.68	63.16	98.25	100.00	100.00	100.00	100.00	100.00	MLP	27.27	27.27	63.64	72.73	81.82	81.82	81.82	81.82
RF	40.43	52.63	63.16	66.67	64.91	66.67	68.42	73.68	RF	27.27	27.27	45.45	45.45	45.45	45.45	45.45	45.45
NB	70.21	50.88	78.95	92.98	92.98	96.49	96.49	98.25	NB	27.27	54.55	81.82	81.82	81.82	81.82	81.82	81.82
AB	51.06	59.65	87.72	94.74	100.00	100.00	100.00	100.00	AB	27.27	36.36	45.45	54.55	54.55	54.55	54.55	63.64
LB	38.30	56.14	85.96	96.49	100.00	100.00	100.00	100.00	LB	27.27	36.36	45.45	45.45	45.45	63.64	63.64	63.64
BG	40.43	43.86	89.47	100.00	100.00	100.00	100.00	100.00	BG	9.09	27.27	45.45	45.45	45.45	45.45	54.55	90.91
ML tech.	Net								ML tech.	Log4j							
	Original	MC5	MC10	MC15	MC20	MC25	MC30	MC50		Original	MC5	MC10	MC15	MC20	MC25	MC30	MC50
MLP	40.68	79.66	94.92	94.92	94.92	100.00	100.00	100.00	MLP	17.54	63.16	98.25	100.00	100.00	100.00	100.00	100.00
RF	50.85	74.58	83.05	84.75	86.44	89.83	89.83	96.61	RF	26.32	52.63	63.16	66.67	64.91	66.67	68.42	73.68
NB	32.20	52.54	72.88	81.36	88.14	91.53	91.53	93.22	NB	14.04	50.88	78.95	92.98	92.98	96.49	96.49	98.25
AB	61.02	83.05	89.83	96.61	96.61	96.61	96.61	98.31	AB	24.56	59.65	87.72	94.74	100.00	100.00	100.00	100.00
LB	57.63	83.05	94.92	96.61	98.31	96.61	96.61	98.31	LB	26.32	56.14	85.96	96.49	100.00	100.00	100.00	100.00
BG	55.93	79.66	86.44	91.53	93.22	100.00	100.00	100.00	BG	19.30	43.86	89.47	100.00	100.00	100.00	100.00	100.00

Table 18 Precision results of MetaCost learners using ML techniques

ML tech.	Calendar								ML tech.	Bluetooth							
	Original	MC5	MC10	MC15	MC20	MC25	MC30	MC50		Original	MC5	MC10	MC15	MC20	MC25	MC30	MC50
MLP	76.39	25.00	15.46	17.00	17.00	17.00	17.00	17.00	MLP	50.00	38.10	30.00	25.71	24.39	23.26	22.22	19.23
RF	80.16	24.14	16.22	17.50	17.07	15.66	15.48	15.12	RF	33.33	47.06	27.59	29.03	27.27	26.47	25.71	25.00
NB	52.70	25.00	22.86	16.44	17.00	17.00	17.00	17.00	NB	50.00	53.33	42.11	40.00	38.10	38.10	39.13	40.00
AB	76.38	46.67	16.33	16.16	16.16	17.00	17.00	17.00	AB	14.29	40.00	32.00	30.77	29.63	30.00	28.13	25.00
LB	76.22	36.84	17.05	16.49	16.16	16.16	16.16	17.00	LB	33.33	37.50	34.78	27.59	27.59	23.53	21.62	20.93
BG	78.21	26.32	17.00	17.00	17.00	17.00	17.00	17.00	BG	50.00	40.00	31.03	28.13	21.43	16.36	16.13	16.92
ML tech.	MMS								ML tech.	IO							
	Original	MC5	MC10	MC15	MC20	MC25	MC30	MC50		Original	MC5	MC10	MC15	MC20	MC25	MC30	MC50
MLP	56.76	56.76	40.91	38.94	39.13	39.13	39.66	38.98	MLP	100.00	33.33	17.95	13.79	12.86	12.16	10.11	8.57
RF	47.50	47.50	41.90	40.00	89.8	39.64	39.29	38.33	RF	27.27	14.29	20.00	17.24	16.13	14.71	12.82	11.11
NB	47.14	47.14	42.86	43.01	41.67	42.86	42.86	45.65	NB	42.86	13.95	15.79	15.52	14.75	14.06	13.43	12.33
AB	45.28	45.28	87.76	39.47	39.13	38.98	38.33	36.22	AB	37.50	25.00	26.32	20.69	14.29	13.64	13.33	11.67
LB	52.94	52.94	40.57	37.29	36.00	35.71	36.51	33.58	LB	50.00	30.77	20.00	14.71	10.20	11.86	10.45	6.54
BG	67.86	67.86	41.12	38.46	38.14	39.50	38.52	32.64	BG	100.00	27.27	29.41	20.00	16.13	11.36	8.33	6.02
ML tech.	Net								ML tech.	Log4j							
	Original	MC5	MC10	MC15	MC20	MC25	MC30	MC50		Original	MC5	MC10	MC15	MC20	MC25	MC30	MC50
MLP	66.67	46.08	43.41	37.58	34.15	32.24	32.24	29.95	MLP	62.50	29.03	20.90	20.00	20.00	20.00	20.00	20.00
RF	57.69	47.83	43.75	42.37	41.13	40.77	40.77	42.22	RF	44.12	27.03	23.23	21.97	20.56	20.43	20.10	19.27
NB	61.29	60.78	53.75	46.15	45.22	42.52	42.52	42.31	NB	47.06	28.16	19.82	19.78	19.63	20.15	19.71	20.00
AB	61.02	44.55	39.26	39.04	38.51	38.00	38.00	38.16	AB	66.67	25.19	22.62	19.15	20.00	20.000	20.00	20.00
LB	64.15	43.36	41.79	39.58	39.46	37.75	37.75	29.74	LB	68.18	30.19	22.17	20.07	20.00	20.00	20.00	20.00
BG	63.46	43.12	39.53	38.85	37.67	32.78	32.78	29.35	BG	61.11	26.88	21.43	20.00	20.00	20.00	20.00	20.00

Table 19 G-Mean results of MetaCost learners using ML techniques

ML tech.	Calendar								ML tech.	Bluetooth							
	Original	MC5	MC10	MC15	MC20	MC25	MC30	MC50		Original	MC5	MC10	MC15	MC20	MC25	MC30	MC50
MLP	0.80	0.46	0.23	N.D.	N.D.	N.D.	N.D.	N.D.	MLP	0.66	0.60	0.53	0.49	0.48	0.47	0.46	0.42
RF	0.65	0.46	0.36	0.39	0.38	0.35	0.34	0.33	RF	0.53	0.66	0.51	0.52	0.50	0.49	0.48	0.48
NB	0.59	0.46	0.44	0.36	N.D.	N.D.	N.D.	N.D.	NB	0.66	0.70	0.63	0.61	0.60	0.59	0.61	0.62
AB	0.80	0.64	0.29	0.00	0.00	N.D.	N.D.	N.D.	AB	0.34	0.61	0.54	0.53	0.52	0.53	0.51	0.48
LB	0.80	0.57	0.38	0.33	0.00	0.00	0.00	N.D.	LB	0.53	0.59	0.57	0.50	0.50	0.46	0.44	0.44
BG	0.65	0.47	N.D.	N.D.	N.D.	N.D.	N.D.	N.D.	BG	0.65	0.61	0.54	0.51	0.44	0.36	0.33	N.D.
ML tech.	MMS								ML tech.	IO							
	Original	MC5	MC10	MC15	MC20	MC25	MC30	MC50		Original	MC5	MC10	MC15	MC20	MC25	MC30	MC50
MLP	0.68	0.68	0.63	0.61	0.61	0.61	0.62	0.62	MLP	0.98	0.56	0.42	0.37	0.36	0.35	0.31	0.29
RF	0.61	0.61	0.63	0.62	0.93	0.61	0.61	0.61	RF	0.51	0.37	0.44	0.41	0.39	0.38	0.35	0.33
NB	0.64	0.61	0.62	0.63	0.62	0.63	0.63	0.65	NB	0.64	0.37	0.39	0.39	0.38	0.37	0.36	0.34
AB	0.61	0.61	0.91	0.62	0.61	0.62	0.6	0.60	AB	0.59	0.49	0.50	0.45	0.37	0.36	0.36	0.34
LB	0.65	0.65	0.62	0.59	0.59	0.58	0.60	0.57	LB	0.69	0.54	0.44	0.38	0.31	0.34	0.32	0.25
BG	0.74	0.74	0.63	0.61	0.61	0.63	0.62	0.57	BG	0.97	0.51	0.53	0.44	0.39	0.33	0.28	0.24
ML tech.	Net								ML tech.	Log4j							
	Original	MC5	MC10	MC15	MC20	MC25	MC30	MC50		Original	MC5	MC10	MC15	MC20	MC25	MC30	MC50
MLP	0.72	0.64	0.64	0.59	0.56	0.57	0.57	0.55	MLP	0.79	0.50	0.44	N.D.	N.D.	N.D.	N.D.	N.D.
RF	0.68	0.64	0.62	0.61	0.61	0.61	0.61	0.64	RF	0.89	0.48	0.44	0.30	0.41	0.41	0.40	0.39
NB	0.68	0.70	0.68	0.64	0.64	0.63	0.63	0.63	NB	0.74	0.49	0.40	0.39	0.38	0.41	0.36	0.40
AB	0.72	0.63	0.60	0.61	0.61	0.60	0.60	0.61	AB	0.76	0.46	0.45	0.00	N.D.	N.D.	N.D.	N.D.
LB	0.73	0.62	0.63	0.62	0.62	0.60	0.60	0.50	LB	0.81	0.51	0.44	0.41	N.D.	N.D.	N.D.	N.D.
BG	0.72	0.61	0.59	0.60	0.59	0.57	0.57	N.D.	BG	0.85	0.47	0.43	N.D.	N.D.	N.D.	N.D.	N.D.

Table 20 Balance results of MetaCost learners using ML techniques

ML tech.	Calendar								ML tech.	Bluetooth							
	Original	MC5	MC10	MC15	MC20	MC25	MC30	MC50		Original	MC5	MC10	MC15	MC20	MC25	MC30	MC50
MLP	41.76	48.48	29.65	29.29	29.29	29.29	29.29	29.29	MLP	48.42	74.28	69.64	63.61	58.9	56.31	53.72	44.63
RF	33.44	54.38	43.23	42.40	40.74	38.09	37.27	35.62	RF	47.98	77.40	69.08	68.45	66.04	64.83	63.61	62.38
NB	49.73	48.48	56.06	44.02	29.29	29.29	29.29	29.29	NB	54.70	78.65	75.93	75.12	74.28	74.28	77.61	79.33
AB	41.76	57.85	30.02	29.17	29.17	29.29	29.29	29.29	AB	35.24	75.12	70.55	69.55	68.52	69.64	67.25	62.38
LB	41.76	57.17	37.25	30.87	29.17	29.17	29.17	29.29	LB	47.98	76.52	72.47	66.41	66.41	60.87	57.41	53.66
BG	41.71	48.68	29.29	29.29	29.29	29.29	29.29	29.29	BG	35.70	75.12	70.83	67.25	54.92	38.41	31.61	29.29
ML tech.	MMS								ML tech.	IO							
	Original	MC5	MC10	MC15	MC20	MC25	MC30	MC50		Original	MC5	MC10	MC15	MC20	MC25	MC30	MC50
MLP	59.87	59.87	63.40	61.02	60.60	60.60	60.69	59.57	MLP	48.57	48.52	71.25	72.15	72.32	70.89	65.38	59.34
RF	56.26	56.26	65.47	62.69	93.00	62.13	61.57	58.44	RF	48.47	48.07	60.6	60.24	60.04	59.71	59.08	58.22
NB	70.42	63.7	68.43	68.45	66.86	67.69	67.69	69.59	NB	48.55	64.59	76.82	76.49	75.47	74.43	73.39	71.25
AB	61.76	61.76	91.33	61.16	60.60	59.57	58.44	54.52	AB	48.53	54.74	61.02	66.56	64.75	64.42	64.24	66.61
LB	55.46	55.46	64.14	58.23	55.00	54.44	52.84	48.91	LB	48.56	54.86	60.6	59.71	57.57	66.87	64.75	52.30
BG	57.57	57.57	64.36	59.48	58.92	59.59	57.91	45.56	BG	35.72	48.47	61.13	60.60	60.04	58.37	58.33	37.00
ML tech.	Net								ML tech.	Log4j							
	Original	MC5	MC10	MC15	MC20	MC25	MC30	MC50		Original	MC5	MC10	MC15	MC20	MC25	MC30	MC50
MLP	57.63	69.07	63.47	53.55	46.10	38.25	38.25	31.28	MLP	41.67	62.27	34.24	29.29	29.29	29.29	29.29	29.29
RF	63.56	70.09	66.42	64.46	62.41	60.99	60.99	61.09	RF	47.57	58.13	54.83	51.95	49.18	48.40	47.00	42.33
NB	51.69	65.00	73.41	69.15	67.53	63.16	63.16	62.35	NB	39.15	58.37	41.63	33.14	32.52	32.35	30.49	30.52
AB	70.15	67.35	58.54	55.62	54.62	53.63	53.63	53.18	AB	46.61	57.63	46.26	29.19	29.29	29.29	29.29	29.29
LB	68.58	65.95	60.99	56.61	55.67	53.13	53.13	31.77	LB	47.85	61.42	45.74	32.04	29.29	29.29	29.29	29.29
BG	67.43	65.94	59.99	57.25	54.43	39.75	39.75	29.29	BG	42.89	55.05	41.53	29.29	29.29	29.29	29.29	29.29

Table 21 AUC results of MetaCost learners using ML techniques

ML tech.	Calendar								ML tech.	Bluetooth							
	Original	MC5	MC10	MC15	MC20	MC25	MC30	MC50		Original	MC5	MC10	MC15	MC20	MC25	MC30	MC50
MLP	0.53	0.54	0.41	0.42	0.42	0.42	0.42	0.42	MLP	0.79	0.81	0.82	0.81	0.76	0.78	0.73	0.66
RF	0.61	0.52	0.44	0.47	0.48	0.49	0.49	0.48	RF	0.70	0.78	0.76	0.80	0.79	0.77	0.77	0.76
NB	0.58	0.62	0.61	0.54	0.56	0.59	0.59	0.59	NB	0.85	0.87	0.86	0.87	0.86	0.86	0.86	0.87
AB	0.43	0.66	0.44	0.47	0.48	0.5	0.50	0.50	AB	0.66	0.76	0.70	0.74	0.72	0.73	0.72	0.69
LB	0.49	0.61	0.43	0.46	0.39	0.5	0.50	0.51	LB	0.67	0.78	0.73	0.76	0.75	0.66	0.69	0.69
BG	0.54	0.56	0.50	0.50	0.50	0.50	0.50	0.50	BG	0.77	0.77	0.78	0.73	0.63	0.48	0.50	0.50
ML tech.	MMS								ML tech.	IO							
	Original	MC5	MC10	MC15	MC20	MC25	MC30	MC50		Original	MC5	MC10	MC15	MC20	MC25	MC30	MC50
MLP	0.80	0.54	0.70	0.71	0.71	0.71	0.72	0.75	MLP	0.74	0.68	0.69	0.79	0.76	0.79	0.74	0.73
RF	0.61	0.52	0.78	0.73	0.71	0.71	0.72	0.74	RF	0.67	0.64	0.68	0.70	0.68	0.67	0.68	0.69
NB	0.79	0.62	0.77	0.78	0.78	0.78	0.78	0.78	NB	0.72	0.76	0.77	0.77	0.77	0.77	0.76	0.74
AB	0.76	0.66	0.79	0.79	0.80	0.80	0.80	0.80	AB	0.66	0.63	0.68	0.71	0.75	0.75	0.77	0.76
LB	0.79	0.61	0.80	0.76	0.78	0.78	0.81	0.78	LB	0.70	0.68	0.74	0.73	0.75	0.73	0.74	0.70
BG	0.79	0.56	0.71	0.72	0.70	0.70	0.74	0.61	BG	0.67	0.71	0.71	0.66	0.70	0.70	0.58	0.51
ML tech.	Net								ML tech.	Log4j							
	Original	MC5	MC10	MC15	MC20	MC25	MC30	MC50		Original	MC5	MC10	MC15	MC20	MC25	MC30	MC50
MLP	0.77	0.76	0.74	0.75	0.75	0.53	0.51	0.30	MLP	0.67	0.70	0.52	0.36	0.31	0.30	0.29	0.29
RF	0.80	0.79	0.78	0.75	0.73	0.73	0.72	0.72	RF	0.61	0.58	0.53	0.52	0.51	0.51	0.57	0.49
NB	0.78	0.78	0.79	0.79	0.78	0.79	0.77	0.75	NB	0.62	0.61	0.54	0.60	0.60	0.60	0.61	0.62
AB	0.78	0.79	0.79	0.71	0.70	0.72	0.72	0.76	AB	0.66	0.62	0.65	0.50	0.50	0.50	0.50	0.50
LB	0.79	0.78	0.80	0.79	0.77	0.78	0.78	0.50	LB	0.68	0.65	0.66	0.60	0.52	0.52	0.52	0.52
BG	0.78	0.76	0.69	0.66	0.68	0.63	0.54	0.50	BG	0.68	0.60	0.60	0.50	0.50	0.50	0.50	0.50

Table 22 Cost values of MetaCost learners using ML techniques

ML tech.	Calendar							ML tech.	Bluetooth						
	MC5	MC10	MC15	MC20	MC25	MC30	MC50		MC5	MC10	MC15	MC20	MC25	MC30	MC50
MLP	75	102	83	83	83	83	83	MLP	28	41	56	51	58	65	92
RF	72	112	111	128	170	191	273	RF	24	41	52	64	75	86	127
NB	75	117	136	83	83	83	83	NB	22	41	57	73	88	74	65
AB	58	92	98	103	83	83	83	AB	27	47	63	79	71	83	127
LB	62	93	96	103	108	113	83	LB	25	45	66	81	101	119	134
BG	74	83	83	83	83	83	83	BG	27	40	53	73	96	82	54
ML tech.	MMS							ML tech.	IO						
	MC5	MC10	MC15	MC20	MC25	MC30	MC50		MC5	MC10	MC15	MC20	MC25	MC30	MC50
MLP	146	85	114	110	120	100	122	MLP	46	72	95	101	115	140	196
RF	161	91	111	65	142	158	124	RF	58	80	114	146	179	214	340
NB	157	132	158	196	181	206	300	NB	62	68	79	92	105	118	164
AB	144	46	99	110	97	104	131	AB	47	74	98	136	163	189	253
LB	161	103	119	120	131	110	141	LB	44	80	119	164	152	180	300
BG	149	93	102	113	72	75	97	BG	48	72	110	146	189	216	206
ML tech.	Net							ML tech.	Log4j						
	MC5	MC10	MC15	MC20	MC25	MC30	MC50		MC5	MC10	MC15	MC20	MC25	MC30	MC50
MLP	115	103	138	168	124	124	138	MLP	193	222	228	228	228	228	228
RF	123	163	203	233	227	257	178	RF	216	329	420	543	623	695	926
NB	160	197	221	203	198	223	275	NB	214	302	275	297	268	284	274
AB	111	142	119	131	143	153	144	AB	216	241	273	228	228	228	228
LB	114	108	117	109	144	154	187	LB	199	252	249	228	228	228	228
BG	122	158	160	171	121	121	142	BG	228	247	228	228	228	228	228

Table 23 Wilcoxon test results on resample method vs MetaCost learners

Data set	G-Mean	Balance	AUC
Calendar	S+ (0.028)	S+ (0.046)	S+ (0.046)
Bluetooth	S+ (0.028)	S+ (0.046)	S+ (0.027)
MMS	S+ (0.046)	S+ (0.075)	S+ (0.027)
IO	S+ (0.027)	S+ (0.075)	S+ (0.027)
Net	S+ (0.027)	S+ (0.028)	S+ (0.027)
Log4j	S+ (0.027)	S+ (0.046)	S+ (0.028)

should be a researcher's objective to increase both recall as well as precision and both change-prone and not change-prone classes should be correctly identified. For example, result of CR = 5 in Bluetooth and MMS data sets show better or comparable precision values.

After performing the above analysis, it can be seen that the performance of different ML techniques varied significantly using different sampling methods when evaluated using G-mean, balance or AUC performance measure. The mean ranks obtained by the above Friedman tests advocate the resample method with replacement as the best sampling method in majority of the cases. These results of Friedman tests are further supported by the use of Wilcoxon test, according to which the resample with replacement method is significantly better than the other two sampling methods when pairwise comparisons are performed. Furthermore, it was analysed that the performance of different sampling methods was better than the scenario where no sampling method was used.

Table 19 shows the G-mean values with different CR's on all the data sets of the study. It should be noted that there was decrease in G-mean values in four data sets and improvement in G-mean values in only two data sets (Bluetooth and MMS). This is because G-mean is the geometric mean of both accuracies of change prone as well as non-change prone classes. MetaCost learners tend to increase the recall rates i.e. the accuracy of predicting change prone classes which can lead to decrease in the accuracy with which non-change prone classes are predicted. However, this decrease in accurate prediction of non-change prone classes may not be compensated well by the increase in recall values as change prone classes are fewer in number. The decrease in precision values also indicates this trend. Thus, leading to decrease in overall G-mean values for four data sets of the study.

Table 24 Data set details for inter-release validation

Software name	Versions for training	Versions for validation
Android calendar	4.0.2–4.0.4 (35%)	4.0.4–4.1.2 (61%)
Android bluetooth	4.3.1–4.4.2 (16%)	5.0.2–5.1.0 (30%)
Android MMS	2.3.7–4.0.2 (27%)	4.1.2–4.2.2 (52%)
Apache IO	1.3–1.4 (30%)	1.4–2.0 (60%)
Apache net	3.0–3.1 (29%)	3.1–3.2 (51%)
Apache Log4j	1.2.13–1.2.14 (6%)	1.2.16–1.2.17 (25%)

Table 20 shows the balance results of different MetaCost learners on each data set of the study. It was noted that various CR's gave an improved balance results on different data sets. The balance results on Calendar data set was improved up to a maximum of 11%. Similarly, the balance results on Bluetooth, MMS, IO, Net and Log4j data sets were improved up to a maximum of 31, 50, 20, 4 and 15% respectively. Table 21 reports the AUC results of different MetaCost learners. It was noted that certain CR's gave a maximum improvement of 5, 5, 23, 5 and 2% correspondingly in the AUC values of Calendar, Bluetooth, MMS, IO and Log4j data sets respectively. These results favour the use of MetaCost learners as compared to the "original" technique.

We also applied Friedman test on G-mean, balance and AUC values of each data set individually to rank the capability of different CR's along with the "original" technique to evaluate whether the improvement in different performance metrics is statistically significant or not when MetaCost learners are used. The Friedman test was conducted at significance level $\alpha = 0.05$. When Friedman test was conducted on AUC values, four out of six data sets showed that MetaCost learners gave better value than the "original" technique. Out of these four data sets, three gave statistically significant results indicating that MetaCost learners showed statistically significant improvement than the "original" technique. The application of Friedman test on balance values indicated that in five out of six data sets, the MetaCost learners statistically outperformed the "original" technique. However, in only two out of six data sets, the MetaCost learners statistically outperformed the "original" technique on G-mean values. As discussed previously, this decrease in G-mean values is due to decrease in accurate prediction of non-change prone classes which is not compensated by correct prediction of change prone classes as change prone classes are few in number. However, since in majority of cases MetaCost learners gave statistically improved results than the "original" technique, when evaluated using G-mean, balance and AUC performance metrics, they are favourable for learning through imbalanced data.

Upon evaluating MetaCost learners with different CR's, each data set gave a best model with correspondingly different CR. Also, the cost of the model for each CR ratio (shown in Table 22) was considered while choosing the best CR ratio for a specific data set. The Android Bluetooth and Calendar data sets, along with Apache Net and Log4j data sets gave the best MetaCost model with CR = 5. However, the Android MMS data set gave the best performing MetaCost learner with CR = 10 and Apache IO with CR = 15. Thus, we advocate that developers should explore different CR's in order to find the best MetaCost learners. A similar recommendation is given by Seliya and Khoshgoftaar (2011).

Thus, after analysing the results produced by different MetaCost learners, we can see that MetaCost learning is an efficient way to handle imbalanced learning problem. In general, in majority of the cases the results of balance and AUC performance metrics improved statistically as compared to the original learning technique in all the data sets. However, it may be noted that use of different CR's may provide different results. A researcher needs to assess different CR's for a specific data set in order to ascertain the CR which provides the best model.

RQ3 What is the comparative performance of the best sampling method and MetaCost learner for ILP?

According to RQ1, the best sampling method was resample with replacement. This RQ evaluates the predictive performance of change prediction models developed using resample with replacement method with the best MetaCost learner for each data set.

The performance evaluation is done on the basis of G-mean, balance and AUC values of change prediction models by performing Wilcoxon test at the cut-off $\alpha = 0.05$. The test was performed on each data set individually. We tested the following hypothesis H4.

> *Null Hypothesis H4*: Change prediction models developed using various ML Techniques (MLP, RF, NB, AB, LB, BG) do not show significant differences in performance metrics (G-mean, balance and AUC) when Resampling with replacement sampling method is used instead of MetaCost learners.
> *Alternate Hypothesis H4a*: Change prediction models developed using various ML Techniques (MLP, RF, NB, AB, LB, BG) show significant differences in performance metrics (G-mean, balance and AUC) when Resampling with replacement sampling method is used instead of MetaCost learners.

In order to apply Wilcoxon test, we used G-mean, balance and AUC values of all the ML techniques on each data set respectively. Table 23 reports the results of Wilcoxon test and its p-value (shown in parenthesis) when the performance metrics obtained by Resample method are compared with the performance metrics using MetaCost learners. According to the results, the Resample with replacement method significantly out-performed the MetaCost learners in majority of the cases (p-value < 0.05). S+ indicates the significantly better performance of the Resample with replacement method in the specific case. Only in MMS and IO data sets balance values, the results are not significant, though in these two cases too the resample method showed better results than the corresponding MetaCost learner. Thus, we accept the alternate hypothesis, which means that the Resample with replacement method is significantly better than the MetaCost learners.

On comparative analysis of the best sampling method i.e. resample with replacement with MetaCost learners, it was found that the resample with replacement method develops better change prediction models. The performance evaluation was done on the basis of G-mean, balance and AUC performance metrics. The results were statistically analysed using Wilcoxon test. The test supported the resample with replacement technique over MetaCost learners as the sampling method yielded statistically significantly better results than MetaCost learners in majority of the cases on each of the six data sets of the study.

8 Inter-release Validation of Change Prediction Models

This section evaluates the results of inter-release prediction for models which are developed using imbalanced data sets. In order to do so, we first developed training models using a specific release of a data set which is imbalanced in nature. Thereafter, these developed models were validated on a separate release of the corresponding data set. Table 24 reports the data set versions used for training and model development as well as the versions used for model validation. The table also denotes the percentage of changed classes in each data set (shown in parenthesis). The steps for developing inter-release validation models are similar to those for developing ten-fold cross validation models. We first remove outliers from each training data set. It may be noted that the reported percentage of change in training data set is the one which we get after eliminating these outliers. The next step involves application of CFS for reducing the number of independent variables in the training data. After this step, we apply sampling methods or MetaCost learners for developing training models. These training models are

then validated using another release of the same data set, which is specified in Table 24. The validation data will have only those independent variables which were selected by CFS in the corresponding training data set. However, we do not perform any outlier removal in validation data sets.

We now evaluate the results of inter-release validation and discuss the answers to all the three RQ's in light of these results.

> RQ1a: Does the performance of different ML methods in this study significantly improve by using various sampling approaches for ILP in change prediction?
> RQ1b: Which sampling approach performs best amongst different sampling techniques analysed in the study?

After analysing the accuracy results of inter-release validation on various data sets of the study, it was found that in majority of the cases in four data sets (Calendar, MMS, Net and IO), the accuracy values showed an improvement after application of sampling techniques. However, in two data sets (Log4j and Bluetooth), the accuracy values declined but as discussed earlier, accuracy is not an effective indicator for imbalanced data sets (He and Garcia 2009; Gao et al. 2015; Weng and Poon 2008). The recall values showed positive improvement after application of sampling techniques on all the data sets for inter-release validation results as more number of change prone classes were correctly predicted after application of sampling techniques. In four out of six data sets, the precision results decreased after application of sampling techniques. However, in majority of cases, there was only a decrease of 10% in precision values but there was an increase of 30% in recall values in certain cases. Since, the precision values are generally assessed together with recall, the decrease in precision values is not much as compared to improvement of recall values. As discussed by Menzies et al. (2007b), though predictive models should attain both high precision and recall values but in order to optimize one of these measures, the other may have to be compromised especially for imbalanced data sets with low number of positive (change prone) instances. Also, a certain decrease in precision values means that few resources might be wasted in testing classes which are not change prone but are predicted as change prone but critical change prone classes will not be missed by software practitioners as the models exhibit high recall values.

The G-mean results using inter-release validation showed positive improvement when sampling techniques were used in four data sets. However, these results were significant in only two data sets (Net and Log4j), when evaluated using the Friedman test at $\alpha = 0.05$. The balance and AUC results also improved in majority of the data sets when different sampling approaches were applied for handling imbalanced data sets. Four (Calendar, Bluetooth, MMS and Net) out of six data sets showed significant improvement in balance results when Friedman test was applied on balance values of different data sets individually. The other two data sets also showed improved balance results but not significantly. The AUC results obtained using inter-release validation showed positive improvement in five data sets, when a sampling method was used for model development. However, only the IO data set showed a significant improvement in AUC results using Friedman test. Thus, we conclude that the application of sampling methods improve the performance of different ML techniques for handling ILP, when evaluated using G-mean, balance and AUC values. However, these results were significantly better for various

sampling approaches in only 7 out of 18 scenarios as Friedman test was individually applied on each of the six data sets of the study for G-mean, balance and AUC values.

In order to evaluate which sampling method performs best, we analysed the seven significant cases of Friedman test application on G-mean, balance and AUC values of different data sets individually. In four out of these seven significant cases, the resample with replacement method achieved the best rank advocating its better performance as compared to other sampling methods. Thus, our inter-release validation results are comparable to ten-fold cross validation results as Resample with replacement method is advocated as the best sampling method for handling ILP.

> RQ2 What is the effectiveness of different MetaCost learners for improving learning through imbalanced data?

The inter-release validation results were computed on seven different CR's, which were 5, 10, 15, 20, 25, 30 and 50, similar to the manner in which ten-fold cross validation results were computed. These results were compared with the "original" scenario when no cost-sensitive approach was applied. On evaluating the inter-release validation results, we found that the accuracy values showed improvement in only three data sets (Net, IO and MMS). However, in five out of six data sets, the recall values improved drastically. Also, in three (Net, Calendar and IO) out of six data sets, the precision values were improved or remained same. However, there was a decrease in the precision values in the other three data sets. As we discussed earlier, accuracy is not an effective indicator for ILP due to its biased nature and precision values may have to be compromised to achieve high recall values (He and Garcia 2009; Gao et al. 2015; Menzies et al. 2007b; Weng and Poon 2008). Thus, these trends are acceptable.

In most of the data sets, there was an increase in G-mean, balance and AUC values when MetaCost learners were used as compared to the "original" technique, when no MetaCost learner was used. We applied Friedman test at $\alpha = 0.05$ on the values obtained by all ML methods on each of the six data sets of the study. The Friedman test was individually applied on G-mean, balance and AUC results of inter-release validation on each data set of the study. In five out of six data sets, the balance values obtained after using specific CR's significantly outperformed the balance values of the "original" technique. In four out of six data sets, the AUC values obtained using MetaCost learners achieved better Friedman ranks as compared to the "original" techniques. However, the Freidman results were significant in only three data sets when compared using AUC values. When evaluating the G-mean results, it was found that in two data sets the G-mean values obtained using the MetaCost learners significantly outperformed the "original" technique. In all, out of 18 scenarios in which Friedman test was applied, there were 13 cases, when Friedman results were significant. Out of these 13 cases, in nine scenarios, the performance metrics obtained by MetaCost learners significantly outperformed the performance metrics obtained by the "original" technique. Thus, these results advocate the effectiveness of MetaCost learners for improving learning through imbalanced data.

Also, it may be noted that the CR which gave the best change prediction model differed for each specific data set. The Log4j and Calendar data sets gave the best inter-release change prediction models with CR = 5. The best performing MetaCost learner for

Bluetooth and IO data sets was CR = 10. However, the MMS data set obtained the best inter-release change prediction results with CR = 15 and for Net data set, it was CR = 25.

RQ3 What is the comparative performance of the best sampling method and MetaCost learner for ILP?

As indicated in RQ1, the performance of various sampling methods when evaluated using inter-release validation yielded resample with replacement as the best method. In this RQ, we compare the inter-release change prediction models developed using resample with replacement method with the inter-release change prediction models developed using the best MetaCost learner on a specific data set. The comparison was done according to the G-mean, balance and AUC values of change prediction models on each of the six data sets of the study using Wilcoxon test at $\alpha = 0.05$.

The pairwise comparison results of resample method with MetaCost learners on all the six data sets using G-mean values indicated that the performance of MetaCost learners was significantly better than the resample method in two data sets (MMS and Net). However, in Bluetooth data set the G-mean values of resample method were significantly better than the MetaCost learner. In all the other three data sets, though the performance of resample method was better than the MetaCost learners, the results were not significant.

The Wilcoxon test results on the basis of balance values of inter-release change prediction models indicated that in four out of six data sets, the Resample method performed better than the MetaCost learner. However, these results were not significant. The MetaCost learners outperformed the resample method in the other two data sets using balance values, but again these results were not significant.

The results of Wilcoxon test on the AUC values of inter-release change prediction models indicated that the resample method outperformed the MetaCost learners on five data sets of the study. However, the results were significant in only one case. Moreover, in Net data set, the results of resample method and MetaCost learner was equivalent.

The above discussed results indicate that though the Resample with replacement method is better than the MetaCost learners in majority of the cases, the results are not significant. Thus, we conclude that the results of inter-release change prediction models are comparable to that of ten-fold cross validation models as the Resample with replacement method could not significantly outperform the MetaCost learners using G-mean, balance and AUC values of inter-release change prediction models.

9 Threats to Validity

The various threats to validity of the study are discussed in this section.

9.1 Conclusion Validity

Threats to conclusion validity includes sources which may lead to misjudgement of accurate relationship between the dependent and the independent variables (Catal and Diri 2009; Khoshgoftaar et al. 2006). It is also referred to as statistical conclusion

validity. We have been very careful in selection of appropriate statistical tests for analysing the results of the studies. Since the study uses non-parametric tests, they do not require normality pre-conditions to be checked on the underlying data. Moreover, the study uses a widely accepted α value of 0.05 to check proper significance of results. The use of post-hoc tests further increase the confidence in the results of the study. The use of multiple performance evaluators is effective in strengthening the conclusions of the study. Moreover, the use of ten-fold cross validation and ten iterations in the case of resample with replacement method aids in eliminating validation bias of the study.

9.2 Internal Validity

The threat to internal validity associates itself with the "causal effect" of the independent variables on the dependent variable i.e. the change proneness attribute of a class (Zhou et al. 2009). This effect can only be evaluated with the aid of controlled experiments, which are very challenging to actually perform (Briand et al. 2001). Since, evaluation of "causal effect" was not the primary aim of the study, we did not evaluate this effect. Thus, this threat exists in our study. Also, the outliers have been removed for both training as well as validation data, when ten-fold cross validation method have been used for model development. This could also be a possible source of threat to the results of the study. However, since we also ascertain the results on inter-release validation models, where outliers have been only removed from the training data and the results obtained using inter-release validation models are comparable to ten-fold cross validation results, thus this threat is minimized. The study uses ten-fold cross validation with resampling method, where the resampled folds are used for evaluation. This is probable source of threat to the study. However, in inter-release validation, the evaluation is done on different data sets and not resampled folds and the results obtained are comparable to that of ten-fold cross validation. Thus, this threat is also reduced.

9.3 Construct Validity

Construct validity is concerned with the degree to which the independent and dependent variables accurately measure the concepts they represent (Zhou et al. 2009). Since, the study uses the LOC metric, extracted from the change logs using the DCRS tool to evaluate the dependent variable, we are assured of its accuracy. Certain researchers have tried to evaluate the accuracy of various OO metrics selected in the study, which assures us of their correctness (Briand et al. 1998; Briand et al. 1999; Briand et al. 2000). Moreover, all the OO metrics used in the study are widely accepted and used in the software engineering community. Thus, the threat to construct validity is minimized.

9.4 External Validity

The external validity is concerned with the extent to which the results of the study are generalizable. This study uses six popular open source data sets which are developed in Java language for empirical validation. Thus, the results of the study can be widely used

in similar scenarios. Moreover, the study clearly states the parameter settings of all the ML techniques used, which aids the replicability of results.

10 Conclusions and Future Directions

The study developed and analysed change prediction models using six imbalanced software data sets. In order to deal with imbalanced learning problem, the study used three data sampling methods (Resample with replacement, Spread subsample and SMOTE) and MetaCost learners using seven different cost ratios. The change prediction models were developed using six ML techniques. The developed models were validated using two approaches: ten-fold cross validation and inter-release validation and were assessed using three unbiased and robust performance metrics namely G-mean, Balance and AUC. The trend of traditional metrics like accuracy, recall and precision was also observed on models developed by using different techniques for imbalanced learning. The main contribution of the study is to advocate the use of different methods for handling imbalanced data sets as use of such methods leads to vast improvement in results of the change prediction models. The study also involved statistical analysis of the results produced in order to strengthen the conclusions of the study. The conclusions of the study are reported as follows:

- The performance of various ML techniques significantly improved after use of different sampling methods. Moreover, the resample method with replacement gave the best results when evaluated in terms of G-mean, balance and AUC performance metrics. However, the other two sampling methods i.e. SMOTE and spread subsample also showed good results.
- The use of MetaCost learners is yet another efficient way of handling imbalanced learning problem as they sensitize the ML method by providing different costs for different kinds of misclassification errors so that the total overall cost of misclassification is decreased and the performance of the developed change prediction model improves. However, one should evaluate different cost ratios on a specific data set to identify the best cost ratio that works well on a specific data set.
- A comparative performance of the best sampling method i.e. resample with replacement with the MetaCost learner having the most effective cost ratio on all the data sets reveal that the sampling method significantly outperforms the MetaCost learners when evaluated on the basis of G-mean, balance and AUC performance metrics on ten-fold cross validation models. The results were supported by inter-release validation models. This result advocates that providing effective training data using resampling is a better approach for handling imbalanced learning rather than cost-sensitizing the learners.

Thus, the results of the study can be used by software practitioners and researchers to develop effective change prediction models when faced with imbalanced data sets. Moreover, such models can be built using optimum costs. Future work may include replicated studies with data sets from other domains and programming languages. Furthermore, apart from ML techniques, we may evaluate whether the discussed

approaches for handling imbalanced data sets are effective when search-based techniques are used for model development.

References

Apandi ZFM, Mustapha N, Affendey LS (2011) Evaluating integrated weight linear method to class imbalanced learning in video data. In 3rd Conference on *Data Mining and Optimization,* 243–247

Arisholm E, Briand LC, Johannessen EB (2010) A systematic and comprehensive investigation of methods to build and evaluate fault prediction models. J Syst Softw 83(1):2–17

Bekkar M, Djemaa HK, Alitouche TA (2013) Evaluation measures for models assessment over imbalanced data sets. J Inf Eng Appl 3(10):27–38

Bieman J, Jain D, Yang H (2001) OO design patterns, design structure, and program changes: an industrial case study. In proceedings of 17th International Conference on Software Maintenance, 580–589

Breiman L (1996) Bagging predictors. Mach Learn 24:123–140

Briand L, Daly J, Wust J (1998) A unified framework for cohesion measurement in object-oriented systems. Empir Softw Eng 3(1):65–117

Briand L, Daly J, Wust J (1999) A unified framework for coupling measurement in object-oriented systems. IEEE Trans Softw Eng 25(1):91–121

Briand L, Wust J, Daly JW (2000) Exploring the relationship between design measures and software quality in object-oriented systems. J Syst Softw 51(3):245–273

Briand L, Wust J, Lounis H (2001) Replicated case studies for investigating quality factors in object oriented designs. Empir Softw Eng J 6(1):11–58

Brieman L (2001) Random forests. Mach Learn 45(1):5–32

CartWright M, Shepperd M (2000) An empirical investigation of an object-oriented software system. IEEE Tran Softw Eng 26(8):786–796

Carvalho ABD, Pozo A, Vergilio SR (2010) A symbolic fault-prediction model based on multi-objective particle swarm optimization. J Syst Softw 83(5):868–882

Catal C, Diri B (2009) Investigating the effect of dataset size, metrics sets, and feature selection techniques on software fault prediction problem. Inf Sci 179(8):1040–1058

Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP (2002) SMOTE: synthetic minority over-sampling technique. J Artif Intell Res 16:321–357

Chidamber SR, Kemerer CF (1994) A metrics suite for object oriented design. IEEE Tran Softw Eng 20(6):476–493

Demšar J (2006) Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res 7:1–30

Domingos P (1999) Metacost: a general method for making classifiers cost-sensitive. In Proc. of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, CA, 155–164

Elish MO, Al-Khiaty MA (2013) A suite of metrics for quantifying historical changes to predict future change-prone classes in object-oriented software. J Softw: Evol Process 25(5):407–437

Eski S, Buzluca F (2011) An empirical study on object-oriented metrics and software evolution in order to reduce testing cost by predicting change prone classes. In Proc. of International Conference on Software Testing, Verification and Validation Workshop, 566–571

Fawcett T (2006) An introduction to ROC analysis. Pattern Recogn Lett 27(8):861–874

Friedman J, Hastie T, Tibshirani R (2000) Additive logistic regression: a statistical view of boosting. Ann Stat 28(2):337–407

Galar M, Fernandez A, Barrenechea E, Bustince H, Herrera F (2012) A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. IEEE Trans Syst Man Cybern Part C Appl Rev 42(4):463–484

Gao K, Khoshgoftaar TM, Napolitano A (2015) Combining feature subset selection and data sampling for coping with highly imbalanced software data. In Proc. of 27th International Conf. on Software Engineering and Knowledge Engineering, Pittsburgh, 2015

Giger E, Pinzger M, Gall HC (2012) Can we predict type of code changes? An empirical analysis. In Proc. of 9th IEEE Working Conference on Mining Software Repositories, 217–226

Hall MA (2000) Correlation-based feature selection for discrete and numeric class machine learning. In Proc. of the Seventeenth International Conference on Machine Learning, 359–366

Hall MA, Holmes G (2003) Benchmarking attribute selection techniques for discrete class data mining. IEEE Trans Knowl Data Eng 15(6):1437–1447

Harman M, Islam S, Jia Y, Minku LL, Sarro F, Sirivisut K (2014) less is more: temporal fault predictive performance over multiple Hadoop releases. In Proc. 6th International Symposium on Search Based Software Engineering, 240–246

Haykin S (2004) Neural networks: a comprehensive foundation, 2nd edn. Pearson education, Delhi

He H, Garcia EA (2009) Learning from imbalanced data. IEEE Trans Knowl Data Eng 21(9):1263–1284

Henderson-Sellers B (1996) Object-oriented metrics, measures of complexity. Prentice Hall

Hirohisa AMAN, Mochiduki N, Yamada H (2006) A model for detecting cost-prone classes based on Mahalanobis-Taguchi method. IEICE Trans Inf Syst 89(4):1347–1358

Hulse JV, Khoshgoftaar TM, Napolitano A, Wald R (2009) Feature selection with high-dimensional imbalanced data. In Proc. of International Conference on Data Mining Workshops, 507–514

Jeni L, Cohn JF, De La Torre F (2013) Facing imbalanced data–recommendations for the use of performance metrics. In Proc. of Humane Association Conf. on Affective Computing and Intelligent Interaction, 245–251

Kamei Y, Monden A, Matsumoto S, Kakimoto T, Matsumoto K (2007) The effects of over and under sampling on fault-prone module detection. In Proc. 1st International Symposium on Empirical Software Engineering and Measurement, 196–204

Khoshgoftaar TM, Seliya N, Sundaresh N (2006) An empirical study of predicting faults with case-based reasoning. Softw Qual J 14(2):85–111

Koru AG, Liu H (2007) Identifying and characterizing change-prone classes in two large-scale open-source products. J Syst Softw 80:63–73

Koru AG, Tian J (2005) Comparing high-change modules and modules with the highest measurement values in two large-scale open-source products. IEEE Trans Softw Eng 31(8):625–642

Kubat M, Matwin S (1997) Addressing the curse of imbalanced training sets: one sided selection. In Proc. of 14th International Conference on Machine Learning 97: 179–186

Lessmann S, Baesans B, Mues C, Pietsch S (2008) Benchmarking classification models for software defect prediction: a proposed framework and novel finding. IEEE Trans Softw Eng 34(4):485–496

Li M, Zhang H, Whu R, Zhou Z (2012) Sample-based software defect prediction with active and semi-supervised learning. Autom Softw Eng 19(2):201–230

Liu Y, An A, Huang X (2006) Boosting prediction accuracy on imbalanced datasets with SVM ensembles. In Advances in Knowledge Discovery and Data Mining, 107–118

Lopez V, Fernandez A, Garcia S, Palade V, Herrera F (2013) An insight into classification with imbalanced data: empirical results and current trends on using data intrinsic characteristics. Inf Sci 250:113–141

Lu H, Zhou Y, Xu B, Leung H, Chen L (2012) The ability of object-oriented metrics to predict change-proneness: a meta-analysis. Empir Softw Eng J 17(3):200–242

Malhotra R (2015) A systematic review of machine learning techniques for software fault prediction. Appl Soft Comput 27:504–518

Malhotra R, Khanna M (2013) Investigation of relationship between object-oriented metrics and change proneness. Int J Mach Learn Cybern. Springer-Verlag 4(4): 273–286

Malhotra R, Nagpal K, Upmanyu P, Pritam N (2014) Defect collection and reporting system for git based open source software. In Proc. of International Conf. on Data Mining and Intelligent Computing, 1–7

Martin RC (2002) Agile software development: principles, patters, and practices. Prentice Hall, USA

Menzies T, Greenwald J, Frank A (2007a) Data mining static code attributes to learn defect predictors. IEEE Trans Softw Eng 33(1):2–13

Menzies T, Dekhtyar A, Distefance J, Greenwald J (2007b) Problems with precision: a response to comments on 'data mining static code attributes to learn defect predictors'. IEEE Trans Softw Eng 33(9):637–640

Munkhdalai T, Namsrai OE, Ryu KH (2015) Self-training in significance space of support vectors for imbalanced biomedical event data. BMC Bioinf 16(7):1

Murphy KP (2006) Naive Bayes classifiers, Technical Report

Olague H, Etzkorn L, Gholston S, Quattlebaum S (2007) Empirical validation of three software metric suites to predict the fault-proneness of object-oriented classes developed using highly iterative or agile software development processes. IEEE Trans Softw Eng 33(10):402–419

Pai GJ, Dugan JB (2007) Empirical analysis of software fault content and fault proneness using Bayesian methods. IEEE Trans Softw Eng 33(10):675–686

Phua C, Alahakoon D, Lee V (2004) Minority report in fraud detection: classification of skewed data. SIGKDD Explorations 6(1): 50–59

Rodriguez D, Herraiz I, Harrison R, Dolado J, Riquelme JC (2014) Preliminary comparison of techniques for dealing with imbalance in software defect prediction. In Proc. of the 18th International Conf. on Evaluation and Assessment in Software Engineering, 43

Romano D, Pinzger M (2011) Using source code metrics to predict change-prone java interfaces. 27th IEEE International Conference on Software Maintenance, 303–312
Seiffert C, Khoshgoftaar TM, Hulse JV, Folleco A (2014) An empirical study of the classification performance of learners on imbalanced and noisy software quality data. Inf Sci 259:571–595
Seliya N, Khoshgoftaar TM (2011) The use of decision trees for cost-sensitive classification: an empirical study in software quality prediction. Wiley Interdiscip Rev: Data Min Knowl Disc 1:448–459
Shatnawi R (2012) Improving software fault-prediction for imbalanced data. In Proc. of International Conf. on Innovations in Information Technology, 54–59
Singh Y, Kaur A, Malhotra R (2009) Empirical validation of object-oriented metrics for predicting fault proneness models. Softw Qual J 18:3–35
Stone M (1974) Cross-validatory choice and assessment of statistical predictions. J R Soc A 36:111–114
Su CT, Hsiao YH (2007) An evaluation of the robustness of MTS for imbalanced data. IEEE Trans Knowl Data Eng 19(10):1321–1332
Tan M, Tan L, Dara S, Mayeux C (2015) Online defect prediction for imbalanced data. In Proc. of 37th International Conf. on Software Engineering
Visa S, Ralescu A (2005) Issues in mining imbalanced data sets- a review paper. In Proc. of 16th Conference on Artificial Intelligence and Cognitive Science, 67–73
Wang S, Yao X (2013) Using class imbalance learning for software defect prediction. IEEE Trans Reliab 62:434–443
Weiss GM (2004) Mining with rarity: a unifying framework. ACM SIGKDD Explor Newslett 6(1):7–19
Weiss GM, McCarthy K, Zabar B (2007) Cost-sensitive learning vs. sampling: which is best for handling unbalanced classes with unequal error costs?. In Proc. of International Conf. on Data Mining, 35–41
Weng CG, Poon J (2008) A new evaluation measure for imbalanced datasets. In Proc. of the 7th Australian Data Mining Conference, 27–32
Witten IH, Frank E, Hall MA (2011) Data mining: practical machine learning tools and techniques, 3rd edn. Morgan Kaufmann, San Francisco
Xu R, Chen T, Xia Y, Lu Q, Liu B, Wang X (2015) Word embedding composition for data imbalances in sentiment and emotion classification. Cogn Comput 7(2):226–240
Yang P, Yoo PD, Fernando J, Zhou BB, Zhang Z, Zomaya AY (2014) Sample subset optimization techniques for imbalanced and ensemble learning problems in bioinformatics applications. IEEE Trans Cybern 44(3):445–455
Zhang X, Li Y (2011) An empirical study of learning from imbalanced data. In Proc. of the 22nd Australasian Database Conf, 85–94
Zhou Y, Leung H, Xu B (2009) Examining the potentially confounding effect of class size on the associations between object metrics and change proneness. IEEE Trans Softw Eng 35(5):607–623

Dr. Ruchika Malhotra is Associate Head and Assistant Professor at the Department of Software 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, Indian 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 master's 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 20 as reported by Google Scholar.

She is executive editor of Software Engineering: An International Journal. She is author of book titled "Empirical Research in Software Engineering" published by CRC Press, USA and co-author of a book on Object Oriented Software Engineering published by PHI Learning. Her research interests are in empirical research in software engineering, improving software quality, statistical and adaptive prediction models, software metrics, the search-based software engineering and software testing. She has published more than 130 research papers in international journals and conferences. She can be contacted by e-mail at ruchikamalhotra2004@yahoo.com.

Megha Khanna is currently pursuing her doctoral degree from Delhi Technological University. She is currently working in Sri Guru Gobind Singh College of Commerce, University of Delhi. She completed her master's degree in software engineering in 2010 from the University School of Information Technology, Guru Gobind Singh Indraprastha University, India. She received her graduation degree in computer science (Hons.) in 2007 from Acharya Narendra Dev College, University of Delhi. Her research interests are in software quality improvement, applications of machine learning techniques in change prediction, and the definition and validation of software metrics. She has various publications in international conferences and journals. She can be contacted by email at meghakhanna86@gmail.com.