World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:10, No:4, 2016

An Empirical Evaluation of Performance of Machine Learning Techniques on Imbalanced Software Quality Data

Ruchika Malhotra, Megha Khanna

***Abstract*—**The development of change prediction models can help the software practitioners in planning testing and inspection resources at early phases of software development. However, a major challenge faced during the training process of any classification model is the imbalanced nature of the software quality data. A data with very few minority outcome categories leads to inefficient learning process and a classification model developed from the imbalanced data generally does not predict these minority categories correctly. Thus, for a given dataset, a minority of classes may be change prone whereas a majority of classes may be non-change prone. This study explores various alternatives for adeptly handling the imbalanced software quality data using different sampling methods and effective MetaCost learners. The study also analyzes and justifies the use of different performance metrics while dealing with the imbalanced data. In order to empirically validate different alternatives, the study uses change data from three application packages of open-source Android data set and evaluates the performance of six different machine learning techniques. The results of the study indicate extensive improvement in the performance of the classification models when using resampling method and robust performance measures.

***Keywords*—**Change proneness, empirical validation, imbalanced learning, machine learning techniques, object-oriented metrics.

I. Introduction

OVER the years, a number of studies have established the capability of software quality classification models for improving software quality by identifying change-prone and fault-prone modules or classes [1]-[3]. The prediction models learn from historical software quality data and software metrics capturing various constructs (such as coupling, cohesion) are used for developing these models. Various machine learning (ML) techniques are used for developing the metric models. In binary prediction models, the outcome variable (or predictor variable) has two categories. For example, the classes in an object-oriented (OO) system can either belong to change-prone category or to non-change prone category. Software practitioners can benefit from the prediction models by using them for effectively planning testing and inspection resources.

Ruchika Malhotra is working as Assistant Professor with the Department of Software Engineering, Delhi Technological University, Delhi, India. (e-mail: ruchikamalhotra2004@ yahoo.com).

Megha Khanna is currently pursuing her doctoral degree from Delhi Technological University, Delhi India. She is working with Sri Guru Gobind Singh College of Commerce, University of Delhi (e-mail: meghakhanna86@gmail.com).

A number of previous studies have confirmed that software quality data follows the Pareto principle, which states that majority of changes or defects in a software (80%) actually originate from very few classes or modules (20%) [3]. Thus, there are very few change prone or defective classes leading to imbalanced and skewed data. Learning from such imbalanced data is problematic as there are very few instances to effectively train the model, which identifies the property of interest (change prone classes in our case). Thus, the classifier may produce high predictive capability but exhibits extremely poor performance for the minority class [4]. Such models are of no use in practical scenarios.

Although a number of ML techniques have been used in literature for developing software quality models, they may not be effective due to three possible reasons (1) the techniques are generally based on optimizing "accuracy" measure, to which the minority instances contribute little (2) the techniques assume similar class distribution of training and testing data, which may not be the case and (3) the techniques assume uniform costs for all types of misclassification errors, however one type of error may be more critical than the other in order to balance learning [5]. Thus, researchers and practitioners need ways to effectively deal with imbalanced learning problem (ILP) so that the actual performance of software quality prediction models is reflected. Hence, this paper explores the following research questions (RQs): (1) What is the performance of ML techniques for ILP using different sampling methods? (2) Are ML MetaCost learners efficient for developing models using imbalanced data? (3) Which performance metrics should be evaluated for and while dealing with ILP? In order to answer the above RQs, we perform an empirical evaluation on data sets obtained from three application packages of Android operating system software to assess the performance of six ML techniques for predicting change prone nature of classes using OO metrics. The study uses two sampling methods (resampling and SMOTE) along with a robust class of MetaCost learners for dealing with ILP. Also, in order to evaluate the performance of the developed learners, we validate the results of the models using three stable performance metrics. Furthermore, the use of these performance metrics is justified by comparing it with other commonly used performance metrics i.e. "accuracy" and "recall".

The organization of the various sections of the study is as follows: Section II states the related work and Section III discusses the ILP in detail, Section IV describes various

methods used in the study for handling ILP and Section V discusses the performance metrics. Section VI provides the research methodology and Section VII discusses and analyzes the results. Finally, Section VIII provides the conclusion of the study.

II. Related Work

He and Garcia analyzed the nature of ILP and presented a brief review of various state-of-the-art methods for dealing with imbalanced data sets [6]. A study by Weiss discusses in detail the problems of mining rare classes and suggests methods such as cost sensitive learning, sampling etc. to handle them efficiently [7].

According to Zhang and Li, ILP has varied effects on different learning techniques due to unstable effects on the bias and variance elements of errors [8]. Thus, a number of studies in literature have evaluated different sampling techniques for improving software fault prediction models [9]-[12]. Seliya and Khosgoftaar analyzed various cost sensitive learning techniques for handling ILP [13]. Various literature studies analyzed the use of both cost-sensitive and sampling methods and compared these approaches with ensemble learning for ILP [14], [15]. Certain other studies explored the use of feature selection and sampling techniques for learning through imbalanced data sets [16], [17].

Certain studies analyze and recommend the use of various performance metrics for effectively handling imbalanced data sets [18]. Tan et al. compared the performance of resampling techniques and updateable classification on imbalanced data sets for change classification [19].

To the best of author's knowledge, apart from [19], no study has evaluated the effect of imbalanced learning on change classification models. Moreover, [19] does not evaluate the use of MetaCost learners for change prediction. Also, in this study we use stable performance metrics for evaluating the change prediction models, as compared to the use of traditional performance evaluators like precision and recall by many studies in the literature.

III. The Imbalance Data Problem

Efficient handling of imbalanced data sets for developing effective software quality models is critical. An imbalanced data is one that under-represents a specific class with very few examples. This paper deals with two-class imbalance problem, where the minority class is the more important one.

The importance of efficiently handling ILP in the context of software quality can be understood with the help of an example. Consider an OO software quality data set with 1000 instances, where each instance is a collection of OO metrics and indicates whether a class is change prone or not. The objective is to develop a model with high balanced degree of accurate predictions for both change-prone and not change prone classes. Correct identification of change prone classes is crucial as these classes are "weak areas" of a software product. However, the data set is skewed with only 10% of change prone classes and the rest non-change prone in nature (Pareto's Principle). Thus, developing a model using a standard classifier may provide high accuracy, but the accuracy achieved is biased towards the majority (non-change prone) class and the rate of misclassification would be much higher for change-prone instances. Thus, the developed models would exhibit highly accurate predictions (close to 100%) for the non-change prone classes but for change prone classes it would only be 0-10% [6]. This would mean that only 10 out of 100 change-prone classes would be correctly predicted for the software quality data set (assuming 10% accuracy for change-prone classes). This is highly disastrous as incorrectly recognizing a change prone class as not change prone would lead to losses and poor quality software product as classes that require more resources and attention would be deprived of it. Similarly, predicting not change prone classes as change prone would lead to wastage of resources. Thus, we need minimize both type of misclassification errors in order to establish an effective software quality model. A product with high misclassification of change prone classes would be very expensive and has higher probability of budget and schedule overruns as implementation of changes and defects in later phases of a software life cycle is very expensive. An organization which cannot deliver projects on time and within budget gets a bad reputation leading to considerable losses. Moreover, the maintenance effort for such products would be huge as they may be developed with lower quality standards to remain within budget and time restrictions. Thus, it is important for researchers and software quality practitioners to focus their attention on effectively learning from imbalanced data.

IV. Methods for Handling ILP

This section briefly describes the various methods used in the study for effectively handling the ILP. We deal with ILP using two alternatives: (1) Over-Sampling Methods (2) Meta-cost Classifiers.

A. Over-Sampling Methods

An over-sampling approach is the one that oversamples the minority class in order to increase the examples representing the minority class. SMOTE (Synthetic Minority Over-Sampling Technique) method involves over-sampling by creation of "synthetic" examples [20]. In order to over-sample a minority class, k-nearest neighbors are randomly chosen (k=5 in our case). Next, neighbors are selected depending on the amount of over-sampling i.e. 300% over-sampling means three out of five nearest neighbors are chosen and one synthetic sample is created in the direction of each chosen sample. In order to generate the synthetic sample, first the difference between the chosen sample and the nearest neighbor is computed. Next, the difference is multiplied by a random number between 0 and 1 and added to the sample under consideration. These type of generated synthetic samples generalize the decision region of the minority class [20].

The resample technique is used to create a number of random subsamples of the data by using the sampling

technique with or without replacement [21]. This study uses this technique to bias the actual class distribution and produce a uniform class distribution by oversampling the minority class with replacement.

B. MetaCost Classifiers

As discussed previously, it is important to balance both types of misclassification errors. In order to do so, we associate different costs with different types of misclassification errors. MetaCost is a procedure for cost-sensitizing a base classification technique by using another procedure around it which minimizes cost. A misclassification cost is represented by C(m,n), where C is the cost of misclassifying a class 'm' instance as belonging to class 'n'. A MetaCost classifier uses Bayes optimal prediction which is used to minimize the conditional risk (R(m|x)), which is also termed as the expected cost of predicting that an instance 'x' belongs to class 'm', as shown in (1). Here, P(n|x) is the probability that a given instance 'x' belongs to class 'n'. This implies that the instance space 'X' can be partitioned into 'n' regions. These partitions are such that class 'n' is the least-cost (optimal) prediction in region 'n' [22].

$$R(m \mid x) = \sum_{n} P(n \mid x) C(m \mid n) \quad (1)$$

The MetaCost procedure appropriately modifies the class labels of the instances of the training data, such that they are representative of their optimal class. In order to do so, MetaCost uses an ensemble of 'k' classifiers (a variation of Breiman's Bagging with replacement) for learning, and then assigns the class labels to each instance depending on the number of votes it receives or using the probability estimates of 'k' models (k=10). Thus, a new class label is assigned to each instance depending upon two things: the cost ratio and the estimated probability. This new relabeled training set is used by a base classification technique to develop a cost-sensitive prediction model [13], [22].

TABLE I
CONFUSION MATRIX

	Predicted Positive	**Predicted Negative**
Actual Positive	True Positive (TP)	False Negative (FN)
Actual Negative	False Positive (FP)	True Negative (TN)

V. PERFORMANCE METRICS FOR ILP

This study uses two "traditional" performance metrics namely "accuracy" and "recall". Apart from these performance indicators, a number of literature studies have advocated the use G-mean, Area under the Receiver Operating Characteristic (ROC) Curve and Balance for imbalanced data [6], [15]. We first present a confusion matrix for a two-class problem in Table I. In this study positives are change prone classes and negatives are not change prone classes. A brief description of all the performance measures is given in Table II.

TABLE II
DESCRIPTION OF PERFORMANCE METRICS

Performance Metric	**Description**
Accuracy	It is defined as the percentage of correct predictions. $\frac{TP+TN}{TP+FP+FN+TN}*100$
Recall (PD)	It is defined as the percentage of correctly predicted positives amongst actual positives. $\frac{TP}{TP+FN}*100$
Specificity	It is defined as the percentage of correctly predicted negatives amongst actual negatives. $\frac{TN}{FP+TN}*100$
G-mean	It represents high accuracy for both positive as well as negative prediction. $\sqrt{\frac{TP}{TP+FP}*\frac{TN}{TN+FN}}$
Area Under the ROC Curve (AUC)	ROC is defined as a plot between recall values on the vertical axis and 1-specificity values on the horizontal axis. The higher the area under the ROC curve, the better performing model it is.
Balance	It represents the Euclidean distance between a specific pair of (PD., Probability of False Alarm (PF)) to the optimum value of PD =1 and PF =0. PF is the percentage of incorrectly predicted positives amongst actual negatives. $1-\sqrt{\frac{(0-(PF/100))^2+(1-(PD/100))^2}{2}}$ *where* $PF=\frac{FP}{FP+TN}*100$

VI. RESEARCH METHODOLOGY

This section briefly describes the dependent and independent variables, data collection procedure, description of datasets and the various techniques used in the study.

A. Independent and Dependent Variables

This study uses eighteen OO metrics, which illustrate characteristics of an OO software product such as size, cohesion, coupling, etc. These metrics include two commonly used metrics i.e. Chidamber and Kemerer (CK) metric suite [23] and QMOOD metric suite. The CK metric suite includes Weighted Methods of a Class (WMC), Lack of Cohesion in Methods (LCOM), Coupling Between Objects (CBO), Number of Children (NOC), Depth of Inheritance Tree (DIT), and Response for a Class (RFC) metrics. The QMOOD metric suite includes Measure of Aggression (MOA), Data Access Metric (DAM), Method of Functional Abstraction (MFA), Cohesion Among Methods of a Class (CAM) and Number of Public Methods (NPM) metrics. Other OO metrics used were Average Method Complexity (AMC), Coupling Between Methods of a Class (CBM), Inheritance Coupling (IC), LCOM3, Lines of Code (LCO), Afferent Coupling (Ca) and Efferent Coupling (Ce).

The dependent variable used in the study is change-proneness attribute of a class. An OO class is change prone if it is likely to change after the software has been released and goes into operational phase [1]. The variable is binary in nature and has two values i.e. "yes" if a specific class is

change prone in nature or "no" if the class is not change prone. Change is evaluated in terms of LOC added, deleted or modified in a class.

B. Data Collection Procedure and Data Description

The data sets used in the study are collected with the aid of Defect Collection and Reporting System (DCRS) tool [24]. The students of Delhi technological University, India developed the tool. It enables collection of data from software repositories, which use GIT for version controlling. It evaluates the source code of two versions of a software by extracting change logs. Each data point of a data set consists of eighteen independent variables along with change statistics. A binary variable "ALTER" represents the change prone or non-change prone nature of a class.

This study analyzes the source code of three application packages of Android operating system, which is a popular operating system for mobile devices. The source code for the Android system can be downloaded from http://android.googlesource.com. The details of the three android application packages used in the study are given in Table III. It shows the name, versions analyzed, number of data points and the skew ratio of each data set. The skew ratio is defined as the ration of not change prone classes to change prone classes. It should be noted that we analyzed only the Java source code files, ignoring the other media and layout files. Further, we removed all the outliers in each data set using Inter-quartile range filter of the WEKA tool. In order to reduce the independent variables by eliminating noisy and redundant variables, we use Correlation based Feature Selection (CFS) method [25]. After application of the CFS method, the CBO and Ce metrics were found highly correlated to change for Calendar data set. The WMC, RFC, LOC and CAM metrics were extracted for Bluetooth data set and the LCOM3, LOC, DAM, MOA, CAM and AMC metrics for the MMS data set.

TABLE III
DATA SET DESCRIPTION

Application Package	Versions	No. of Data Points	Skew Ratio
Calendar	4.3.1 - 4.4.2	106	4.88
Bluetooth	4.3.1 - 4.4.2	72	4.90
MMS	2.3.7 – 4.0.2	195	2.68

C. Experimental Settings and Techniques

The study evaluates six different ML techniques namely Multilayer Perceptron (MLP), Random Forests (RF), Naïve Bayes (NB), Adaboost (AB), LogitBoost (LB) and Bagging (BG) [21]. We use the default parameter settings for each technique in the WEKA tool. We develop ten-fold cross validation models ten times using each technique and state the average of ten runs.

In order to effectively validate the results of SMOTE technique we used SMOTE with 100%, 200%, 300%, 400% and 500% oversampling. However, due to space constraints we only discuss the results of SMOTE with 100%, 400% and 500% oversampling. For the MetaCost learners, we do not assign any cost to correct predictions i.e. TPs and TNs. However, we penalize the FPs and FNs. As it is more important to predict change prone classes correctly, we penalize the FNs, i.e. the cost of FN is higher than the cost of FP. This will increase the recall values as the model will predict low FNs as they are associated with higher costs. A cost ratio (CR) is defined as the Cost of FN/Cost of FP. This study analyzes three different CRs that are 5, 10 and 15.

VII. Analysis and Results

This section analyzes and states the results of all the six ML techniques using different methods for handling ILP using the three data sets of the study. Tables IV-VI state the average of all the performance metrics for the ten iterations. Each table states the results of the original technique i.e. without any method to encounter ILP, the results of technique with SMOTE i.e. with 100%, 400% and 500% oversampling (SM100, SM400, SM500), with resampling and with MetaCost classification. Tables IV-VI state the results of three CRs i.e. 5, 10 and 15 for MetaCost classification (MC5, MC10, MC15). The metric values for certain cases are "Not Defined" and are depicted by ND in the tables as the formula for these values turned out to be divided by zero. The results show that the value of AUC, recall, G-Mean and balance increased in most of the cases after application of a method for handling ILP. However, the results of accuracy measure generally declined with the use of any method for handling ILP. We present a description and comparative analysis of all the results while answering the RQs of the study.

RQ1: What is the performance of ML techniques for ILP using different sampling methods?

The performance of various ML techniques improved significantly using different sampling methods. Figs. 1-3 depict the percentage improvement of the performance of various ML techniques in terms of AUC and Balance measure for all the three data sets used in the study. The figures represent the percentage improvement of AUC (bars) and the percentage improvement of balance (lines). The percentage improvement is a result of comparing the performance metrics of the "original" technique with the performance metrics achieved after using sampling methods. It may be noted from the figures that the best results were achieved by the resampling method in majority of the cases. The SMOTE method also performed well in almost all cases. The results of SMOTE 400 were best when compared with other SMOTE variations in Calendar and Bluetooth data sets. However, the SMOTE 500 method gave optimum results for MMS data set. These variations in results are due to variation in skewness of the data sets. A researcher should evaluate different combinations of SMOTE, to achieve a balance between both the majority and minority classes.

RQ2: Are MetaCost learners efficient for developing models using imbalanced data?

MetaCost learners improve the efficiency of the predicted models for ILP. Since these techniques take into account the varying misclassification costs, they are sensitive to class distributions and the models are developed in such a way that the total cost is minimized. In order to ascertain the

effectiveness of the MetaCost learners on imbalanced data sets Figs. 4-6 depict the percentage improvement in terms of AUC in all the three data sets respectively (represented by bars in the figure). Again, the percentage improvement is calculated by comparing the AUC of the original technique with AUC of the techniques using different MetaCost learners. The figures also depict the total cost of the developed model (represented by lines). The total cost for a corresponding model is calculated by adding the misclassification costs of FP and FN prediction as in (2). The figures show positive improvement in terms of AUC with the application of MetaCost learners. However, the Calendar and Bluetooth data sets performed most effectively with CR=5, but the MMS data sets achieved best results with CR=10. This could be due to difference in skew ratios of the data sets.

$$Total\ \text{Cost} = \text{Cost of FN} * \text{FN} + \text{Cost of FP} * \text{FP} \quad (2)$$

RQ3: Which performance metrics should be evaluated for and while dealing with ILP?

It can be seen that the results of the accuracy measure decreased in most of the cases after using any method for handling ILP. This is because accuracy is not termed as a good performance metric for imbalanced data sets [6], [17], [26]. The results of accuracy measure using the "original" technique are highly optimistic. They only take into account the high number of not change prone (majority class) classes predicted correctly. The percentage of correctly predicted change prone classes is very low. This can be seen from the recall values. The values of recall performance metric increased after application of any method for handling ILP. However, in this case recall is biased towards the "minority" class.

TABLE IV
AVERAGE RESULTS OF CALENDAR DATA SET

Technique	Performance Metrics	Original	SM100	SM400	SM500	Resample	MC5	MC10	MC15
	Accuracy	85.00	74.36	56.55	51.89	71.85	73.00	16.00	17.00
	Recall	17.65	17.65	35.29	51.96	56.73	29.41	88.24	100
MLP	**G-Mean**	0.80	0.75	0.58	0.52	0.73	0.46	0.23	ND
	AUC	0.53	0.49	0.63	0.56	0.74	0.54	0.41	0.41
	Balance	41.76	41.74	51.74	51.88	66.36	48.48	29.65	29.29
	Accuracy	83.00	76.07	73.21	76.76	80.3	68.00	33.00	31.00
	Recall	5.88	29.41	60.00	67.65	72.61	41.18	70.59	82.35
RF	**G-Mean**	0.65	0.74	0.75	0.78	0.81	0.46	0.36	0.39
	AUC	0.61	0.69	0.80	0.82	0.84	0.52	0.44	0.47
	Balance	33.44	49.97	70.20	75.59	75.97	54.38	43.23	42.40
	Accuracy	81.00	72.65	57.74	55.14	62.745	73.00	64.00	34.00
	Recall	29.41	29.41	30.59	30.39	38.69	29.41	47.06	70.59
NB	**G-Mean**	0.59	0.65	0.61	0.60	0.64	0.46	0.44	0.37
	AUC	0.58	0.60	0.56	0.56	0.64	0.62	0.60	0.54
	Balance	49.73	49.62	49.87	49.73	54.36	48.48	56.06	44.02
	Accuracy	85.00	74.36	55.95	56.22	74.10	82.00	17.00	16.00
	Recall	17.65	17.65	40.00	78.43	61.30	41.18	94.12	94.12
AB	**G-Mean**	0.80	0.75	0.57	0.55	0.76	0.64	0.28	0.00
	AUC	0.43	0.54	0.57	0.61	0.78	0.66	0.44	0.47
	Balance	41.76	41.74	53.27	47.47	68.798	57.85	30.02	29.17
	Accuracy	85.00	53.73	70.83	74.59	78.10	78.00	25.00	18.00
	Recall	17.65	17.65	58.82	67.65	65.00	41.18	88.24	94.12
LB	**G-Mean**	0.80	0.59	0.72	0.75	0.81	0.57	0.38	0.33
	AUC	0.49	0.64	0.69	0.76	0.82	0.61	0.43	0.46
	Balance	41.76	41.41	68.54	74.2	71.749	57.17	37.25	30.87
	Accuracy	83.00	75.21	74.40	69.73	77.00	74.00	17.00	17.00
	Recall	17.65	17.65	62.35	68.63	65.21	29.41	100.00	100
BG	**G-Mean**	0.65	0.79	0.76	0.69	0.78	0.47	ND	ND
	AUC	0.54	0.65	0.76	0.79	0.81	0.56	0.50	0.50
	Balance	41.71	41.76	71.78	69.83	72.351	48.68	29.29	29.29

A good prediction model should achieve an optimum balance between recall as well as specificity values, and both these values should be as high as possible. Domination of any of "recall" or "specificity" values leads to an inappropriate model and wastage of resources or higher loses. Though, the value of G-mean decreased in some cases after application of certain methods for handling ILP, but it provides an accurate measure for evaluating the performance of a classifier on imbalanced data set. This is because it is based on the correct predictive accuracy of both the classes i.e. minority as well as majority class. Thus, a high G-mean value indicates a balanced model, which maximizes the correct predictions of both the classes. However, if a model is prejudiced towards a specific class (minority or majority), we are bound to obtain a low G-mean values [11]. Similarly, ROC analysis is effective for evaluating models developed on imbalanced data sets [18]. It is insensitive to data distribution and does not assume any specific cost of misclassification or previous class

probabilities [9], [16]. Since AUC obtained from ROC analysis provides a trade-off between recall and specificity values, it is effective for evaluation of a classifier's performance over different threshold values [12], [17]. It provides a visual comparative analysis and is considered a stable performance measure for evaluating ILP. Recently, the use of balance performance metric has been advocated for ILP [12]. It is an effective indicator as it measures the Euclidean distance from the optimum point on the ROC curve of the actual (PF, PD) values. Thus, it is an appropriate measure for ILP.

A comparison of resampling techniques and the MetaCost learners using the AUC performance metric reveals that the resampling technique gives better results.

TABLE V
AVERAGE RESULTS OF BLUETOOTH DATA SET

Technique	Performance Metrics	Original	SM100	SM400	SM500	Resample	MC5	MC10	MC15
	Accuracy	83.08	81.58	81.65	82.50	88.86	75.38	64.62	56.92
	Recall	27.27	68.18	89.09	89.39	83.21	72.73	81.82	81.82
MLP	**G-Mean**	0.66	0.77	0.82	0.83	0.89	0.59	0.53	0.49
	AUC	0.79	0.85	0.88	0.89	0.88	0.81	0.82	0.81
	Balance	48.42	75.71	80.11	80.19	86.61	74.28	69.64	63.61
	Accuracy	78.46	77.63	82.57	81.67	96.62	81.54	64.06	63.08
	Recall	27.27	54.55	83.64	83.33	98.27	72.73	80.00	81.82
RF	**G-Mean**	0.53	0.72	0.83	0.81	0.96	0.66	0.51	0.52
	AUC	0.70	0.81	0.88	0.88	0.97	0.78	0.76	0.79
	Balance	47.98	66.58	82.53	81.39	96.38	77.4	69.08	68.45
	Accuracy	83.08	81.58	78.90	83.05	86.62	84.62	78.46	76.92
	Recall	36.36	54.55	70.91	80.30	72.01	72.73	72.73	72.73
NB	**G-Mean**	0.66	0.79	0.79	0.83	0.89	0.71	0.63	0.61
	AUC	0.85	0.88	0.90	0.89	0.91	0.87	0.86	0.87
	Balance	54.70	67.43	77.48	83.13	80.17	78.65	75.93	75.12
	Accuracy	75.38	80.26	78.90	79.17	92.92	76.92	69.23	67.69
	Recall	9.09	72.73	80.00	84.85	91.72	72.73	72.73	72.73
AB	**G-Mean**	0.34	0.75	0.79	0.79	0.93	0.61	0.54	0.53
	AUC	0.66	0.80	0.83	0.87	0.94	0.76	0.70	0.74
	Balance	35.24	77.4	78.86	77.63	92.42	75.12	70.55	69.55
	Accuracy	78.46	80.26	86.24	85.00	96.61	73.85	72.31	63.08
	Recall	27.27	59.09	85.45	87.88	97.58	81.82	72.73	72.73
LB	**G-Mean**	0.53	0.76	0.86	0.85	0.96	0.59	0.57	0.50
	AUC	0.67	0.84	0.91	0.89	0.96	0.78	0.73	0.76
	Balance	47.98	70.02	86.22	84.35	96.49	76.52	72.47	66.41
	Accuracy	83.08	82.89	80.73	81.67	84.55	76.92	66.15	61.54
	Recall	9.09	68.18	78.18	83.33	86.55	72.73	81.82	81.82
BG	**G-Mean**	0.65	0.79	0.81	0.81	0.84	0.61	0.54	0.51
	AUC	0.77	0.79	0.88	0.87	0.94	0.77	0.78	0.73
	Balance	35.70	76.17	80.59	81.39	82.53	75.12	70.83	67.25

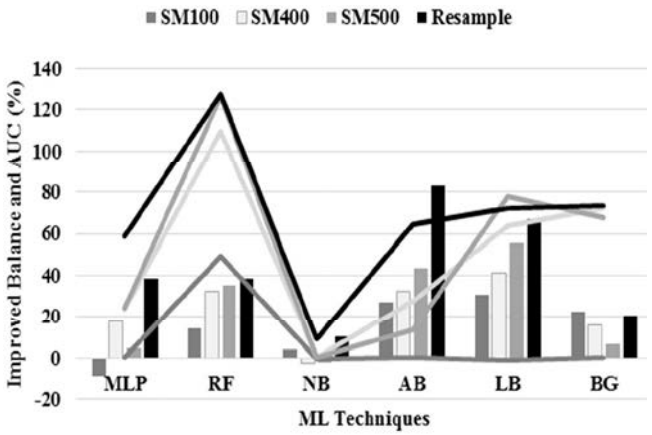

Fig. 1 AUC & Balance Improvement for Calendar using Sampling Methods

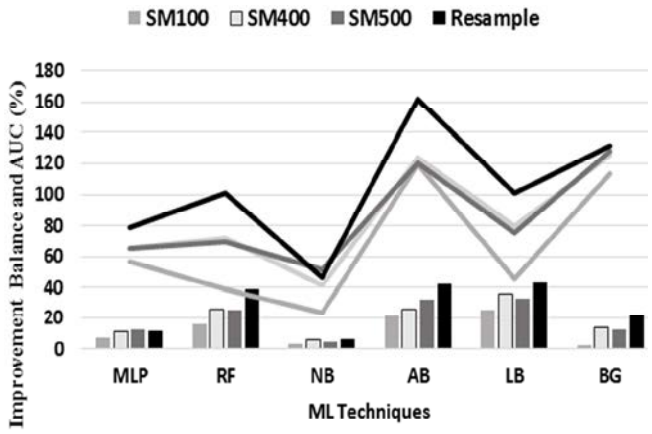

Fig. 2 AUC & Balance Improvement for Bluetooth using Sampling Methods

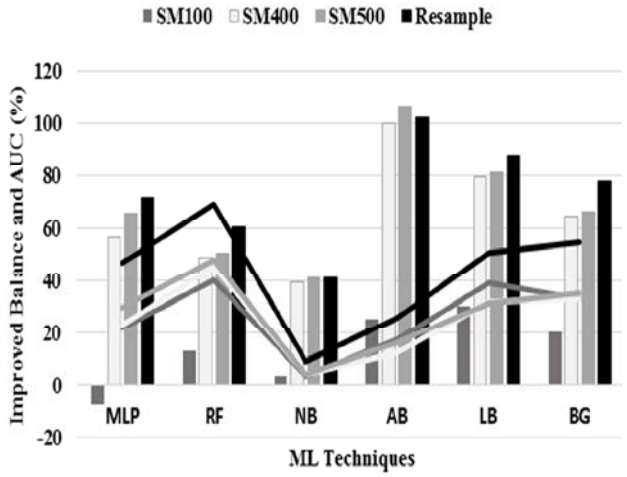

Fig. 3 AUC & Balance Improvement for MMS using Sampling Methods

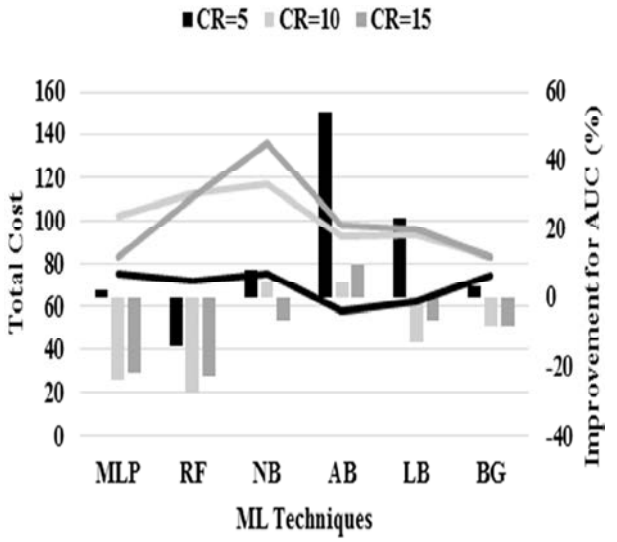

Fig. 4 Total Cost & AUC Improvement for Calendar using MetaCost

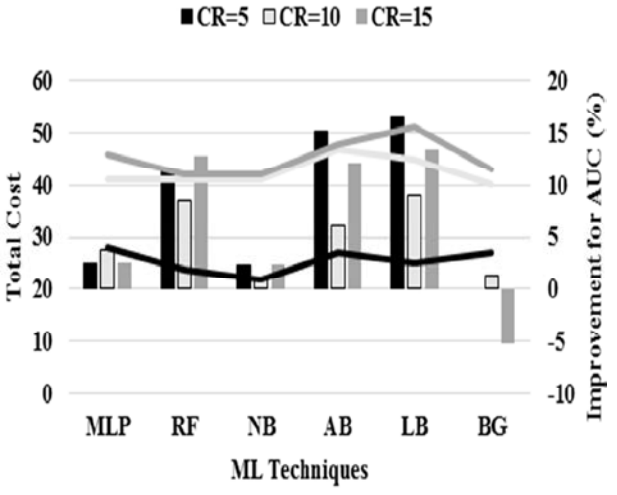

Fig. 5 Total Cost & AUC Improvement for Bluetooth using MetaCost

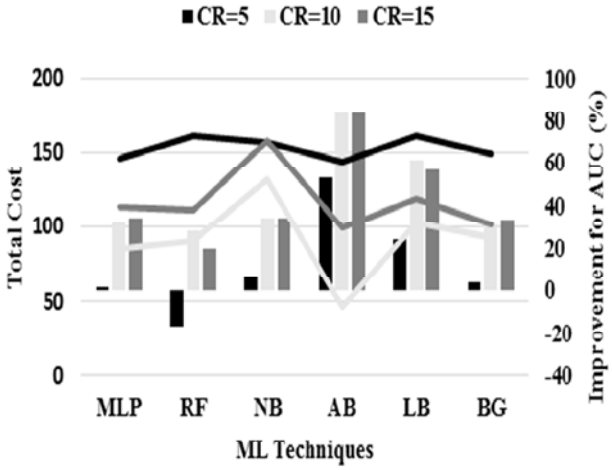

Fig. 6 Total Cost & AUC Improvement for MMS using MetaCost

VIII. Conclusions and Future Work

The goal of this study was to evaluate different methods for handling ILP on change data of three Android data sets. The study evaluated two oversampling techniques (SMOTE and Resample) and MetaCost learners with three different cost ratios using six ML techniques for developing change prediction models using OO metrics. The main contributions of the study include advocating the use of different methods for handling ILP and ascertaining their effectiveness by estimating the improvement in the performance of ML techniques using these methods. Furthermore, the study provides a comprehensive evaluation of the developed models using effective performance metrics. The conclusions of the study are summarized as follows:

- The use of oversampling methods is an effective way of dealing with ILP as considerable improvement is seen in the performance of different ML techniques by their use. The results of the study support the use of resample method for learning on imbalanced data sets. The SMOTE method also provided competent results. However, a researcher should adjust the percentage of over sampling using SMOTE to obtain optimum results, depending upon the skewness of a data set.
- The MetaCost learners are competent and economical methods for handling ILP as their use leads to substantial improvement in a model's performance. Moreover, their capability to perform cost-sensitive classifications results in better models at optimum costs. However, we strongly recommend that a researcher should explore different cost ratios as an important parameter while model development. The cost as well as model's performance should be evaluated while choosing a specific cost ratio parameter setting.
- The study supports the use of stable performance metrics such as ROC analysis, G-mean and Balance for evaluating models developed using imbalanced data. These performance metrics overcome the weakness of traditional metrics like accuracy and recall. They effectively estimate the model's performance without

biasing the results due to presence of a majority class. The resampling technique gave better results than the MetaCost learners using the ROC analysis.

Thus, researchers and practitioners can use the results of the study to develop effective models with optimum associated costs for change prediction using imbalanced data sets. In future, we plan to replicate our study using evolutionary techniques such as genetic algorithms to ascertain their performance using these methods for ILP. Furthermore, we plan to use other software data sets from different domains and programming language environments.

TABLE VI
AVERAGE RESULTS OF MMS DATA SET

Technique	Performance Metrics	Original	SM100	SM400	SM500	Resample	MC5	MC10	MC15
	Accuracy	75.72	74.55	80.89	86.03	88.95	75.72	61.27	58.38
	Recall	44.68	68.09	89.79	93.62	93.71	44.68	95.74	93.62
MLP	**G-Mean**	0.68	0.74	0.80	0.85	0.89	0.68	0.63	0.61
	AUC	0.53	0.49	0.83	0.88	0.91	0.54	0.70	0.71
	Balance	59.87	73.13	73.73	77.65	87.88	59.87	63.40	61.02
	Accuracy	71.68	79.09	88.37	89.95	95.78	71.68	63.01	60.12
	Recall	40.43	77.66	95.74	95.74	97.09	40.43	93.62	93.62
RF	**G-Mean**	0.61	0.79	0.89	0.90	0.96	0.61	0.63	0.62
	AUC	0.61	0.69	0.91	0.92	0.98	0.51	0.78	0.73
	Balance	56.26	78.87	81.79	83.45	95.31	56.26	65.47	62.69
	Accuracy	70.52	72.73	77.84	78.19	78.27	66.67	65.32	65.32
	Recall	70.21	78.72	84.26	83.69	86.62	57.89	82.98	85.11
NB	**G-Mean**	0.64	0.73	0.75	0.74	0.79	0.61	0.62	0.63
	AUC	0.58	0.60	0.81	0.82	0.82	0.62	0.78	0.78
	Balance	70.42	72.98	73.42	73.25	76.73	63.70	68.43	68.45
	Accuracy	69.94	73.18	82.27	85.05	80.29	69.94	91.30	58.96
	Recall	51.06	85.11	95.74	96.10	90.93	51.06	91.49	95.74
AB	**G-Mean**	0.61	0.74	0.84	0.86	0.81	0.61	0.91	0.62
	AUC	0.43	0.54	0.86	0.89	0.87	0.66	0.79	0.79
	Balance	61.76	72.64	69.55	71.80	77.66	61.76	91.33	61.16
	Accuracy	73.99	77.27	81.99	85.05	84.45	73.99	61.27	55.49
	Recall	38.30	84.04	92.34	95.39	88.72	38.30	91.49	93.62
LB	**G-Mean**	0.65	0.77	0.82	0.85	0.85	0.65	0.62	0.59
	AUC	0.49	0.64	0.88	0.89	0.92	0.61	0.79	0.77
	Balance	55.46	77.35	73.07	72.87	83.72	55.46	64.14	58.23
	Accuracy	78.61	77.27	85.04	87.75	89.77	78.61	61.85	57.23
	Recall	40.43	74.47	94.47	96.10	92.32	40.43	93.62	95.74
BG	**G-Mean**	0.74	0.77	0.86	0.88	0.90	0.74	0.63	0.61
	AUC	0.54	0.65	0.89	0.90	0.96	0.56	0.71	0.72
	Balance	57.57	76.79	76.66	77.94	89.29	57.57	64.36	59.48

REFERENCES

[1] R. Malhotra and M. Khanna, "Investigation of Relationship between Object-oriented Metrics and Change Proneness," *Int. J. Mach. Learn. & Cyber.*, vol. 4, 2013, pp. 273-286.

[2] K.K. Aggarwal, Y. Singh, A.Kaur and R. Malhotra, "Empirical Analysis for Investigating the Effect of Object-Oriented Metrics on fault Proneness: A Replicated Case Study," *Software Process: Improvement and Practice*, vol. 16, 2009, no. 1, pp. 39-62.

[3] A.G. Koru and J. Tian, "Comparing High-Change Modules and Modules with the Highest Measurement Values in two Large-Scale Open-Source Products," *IEEE Trans. Softw. Eng.,* vol. 31, 2005, no. 8, pp. 625-642.

[4] M. Kubat and S. Martin, "Addressing the Curse of Imbalanced Data Sets: One Sided Sampling," in *Proc. of 14th International Conf. on Machine Learning*, Nashville, 1997, pp. 179-186.

[5] S. Visa, and A. Ralescu, "Issues in Mining Imbalanced Data Sets- A Review Paper," in *Proc. of the 16th midwest Artificial Intelligence and Cognitive Science Conf.*, 2005, Ohio, pp. 67-73.

[6] H. He, and E.A. Garcia, "Learning from imbalanced data, *IEEE Trans. Knowl. Data Eng.*, vol. 21, 2009, pp. 1263–1284.

[7] G.M. Weiss, "Mining with Rarity: A Unifying Framework.," *ACM SIGKDD Explorations Newsletter*, vol. 6, no. 1. pp. 7-19, 2004.

[8] X. Zhang and Y. Li, "An Empirical Study of Learning from Imbalanced Data." in *Proc. of the 22nd Australasian Database Conf*, Perth, 2011, pp. 85-94.

[9] R. Shatnawi, "Improving Software Fault-Prediction for Imbalanced Data." in *Proc. of International Conf. on Innovations in Information Technology*, Al Ain, 2012, pp. 54-59.

[10] C. Seiffert, T. M. Khoshgoftaar, J. V. Hulse, and A. Folleco, "An Empirical Study of the Classification Performance of Learners on Imbalanced and Noisy Software Quality Data," *Information Sciences*, vol. 259, 2014, pp. 571-595.

[11] Y. Liu, A. An, and X. Huang, "Boosting Prediction Accuracy on Imbalanced Datasets with SVM Ensembles," *In Advances in Knowledge Discovery and Data Mining*, pp. 107-118, 2006.

[12] S. Wang and X. Yao, "Using Class Imbalance Learning for Software Defect Prediction," *IEEE Transactions on Reliability*, vol. 62, 2013, pp. 434-443.

[13] N. Seliya and T. M. Khoshgoftaar, "The Use of Decision Trees for Cost-sensitive Classification: An Empirical Study in Software Quality Prediction," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 1, 2011, 448-459.

[14] G.M.Weiss, K. McCarthy, and B. Zabar, "Cost-sensitive Learning vs. Sampling: Which is Best for Handling Unbalanced Classes with Unequal Error Costs?" in *Proc. of Internatioanl Conf. on Data Mining*, Omaha NE, 2007 pp. 35-41.

[15] D. Rodriguez, I. Herraiz, R. Harrison, J. Dolado, and J. C. Riquelme, "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, London, 2014, p. 43.
[16] J.V. Hulse, T. M. Khoshgoftaar, A. Napolitano, and Randall Wald, "Feature Selection with High-dimensional Imbalanced Data." in *Proc. of International Conf. on Data Mining Workshops*, Florida, 2009, pp. 507-514.
[17] K. Gao, T. M. Khoshgoftaar, and Amri Napolitano, "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.
[18] L. Jeni, J. F. Cohn, and F. De La Torre, "Facing Imbalanced Data--Recommendations for the Use of Performance Metrics." *In Proc. of Humaine Association Conf. on Affective Computing and Intelligent Interaction*, Geneva, 2013, pp. 245-251.
[19] M. Tan, L. Tan, S. Dara, and C. Mayeux, "Online Defect Prediction for Imbalanced Data," in *Proc. of 37th International Conf. on Software Engineering*, Florence, 2015.
[20] N.V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-Sampling Technique," *Journal of Artificial Intelligence Research,* vol. 16, 2002, pp. 321-357.
[21] I.H. Witten and E. Frank, Data Mining: Practical Machine Learning Tools and Techniques, vol. 2, 2005.
[22] P. Domingos, "Metacost: A General Method for Making Classifiers Cost-sensitive," In Proceedings of the 5th ACM SIGKDD *International Conference on Knowledge Discovery and Data Mining*, CA, 1999, pp. 155-164.
[23] S. Chidamber and C. Kemerer, "A Metric Suite for Object- Oriented design," *IEEE Transactions on Software Engineering*, vol. 20, 1994, pp. 476-493.
[24] R. Malhotra, K. Nagpal, P. Upmanyu & N. Pritam, "Defect collection and reporting system for Git based open source software. in *Proc. of International Conf. on Data Mining and Intelligent Computing*, Delhi, 2014, pp. 1-7.
[25] M.A. Hall, "Correlation-based Feature Selection for Discrete and Numeric Class Machine Learning," in *Proc. of the Seventeenth International Conf. on Machine Learning*, CA, 2000, pp. 359-366.
[26] C.G. Weng, and J. Poon, "A New Evaluation Measure for Imbalanced Datasets," in *Proc. of the 7th Australasian Data Mining Conf.*, Sydney, 2008, pp. 27-32.

Ruchika Malhotra is an assistant professor in the Department of Software Engineering, Delhi Technological University (formerly Delhi College of Engineering), Delhi, India. She has been awarded prestigious UGC Raman Postdoctoral Fellowship by the Indian government for pursuing postdoctoral research from the Department of Computer and Information Science, Indiana University-Purdue University Indianapolis (2014-15), Indianapolis, Indiana, USA. 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 was an assistant professor at the University School of Information Technology, Guru Gobind Singh Indraprastha University, Delhi, India. She received the prestigious IBM Faculty Award 2013. She is author of the book titled Empirical Research in Software Engineering: Concepts, Analysis and Applications and co-author of a book Object Oriented Software Engineering. Her research interests are in empirical research in software engineering, improving software quality, statistical and adaptive prediction models, software metrics and software testing. Her H-index as reported by Google Scholar is 17. She has published more than 100 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.