International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878, Volume-8 Issue-4, November 2019

Bug Severity Prediction using Class Imbalance Problem

Shubhra Goyal Jindal, Arvinder Kaur

Abstract*: Class imbalance problem is often observed when instances of major class exceed instances of minor class. The performance of machine learning techniques is immensely afflicted by imbalanced data in several fields. The skewed distribution either predicts the majority class with high error rate or will not foresee the minority class. To solve the problem of imbalanced data of software bugs, Synthetic minority oversampling technique (SMOTE) is used which balances the imbalanced datasets of Apache Projects. It is applied on summary of bugs to balance the dataset and predicts severity at system and component level. Several machine learning techniques are applied on imbalanced as well as balanced datasets to predict the severity of software bugs using textual description. Test outcomes and statistical analysis shows improved results on balanced datasets in respect to Gmean and balance metrics instead of machine learning techniques applied on imbalanced data. Evaluation metrics Gmean improves by 34% and balance by 11% at system level and by 42% and 62% at component level. Further, it was observed that solving class imbalance problem on textual data is helpful in augmenting the performance.

***Keywords*: Class imbalance problem, severity prediction, synthetic minority oversampling technique, Software bug reports**

I. INTRODUCTION

T o develop effective prediction models, adequate training data is required with appropriate number of severe and non-severe bugs in case of severity prediction. However, datasets in real world are imbalance in nature i.e. occurrences of one class massively outnumbers the other class (severe bugs in our case). Most of the classifier algorithms are devised for balanced dataset and poor performance is achieved in imbalance dataset. This problem occurs in several applications namely, social network services, online gaming services, wireless networks, network intrusion detection [1], face recognition [2], financial risk assessment [3], medical diagnosis [4] and many more. It is more prevalent with machine learning including software quality estimation [5,6] software defect prediction [6-16]. The methods to handle class imbalance problem are divided into four classes: data sampling, cost sensitive, kernel based and active learning [17]. All the datasets used in above applications comprise of numeric attributes. In this work, data sampling method is used to predict the severity of software bugs from textual description. Data sampling method is self-sufficient to train the classifier and can be used at data processing stage. Synthetic minority oversampling technique (SMOTE) [18] is used which synthetically generate samples of minority class. We have enforced SMOTE on the textual attribute particularly bug description, which is used to determine the severity of bugs. The bug descriptions of minority class (non-severe in our case) are synthesized. Text mining techniques are applied on one-line bug description and dictionary of frequent occurring terms is formed. Machine learning classifiers are then applied and performance is compared before and after solving the imbalance problem in terms of Geometric mean (G_{mean}) and balance. The approach is evaluated using various techniques namely, naïve bayes, maximum entropy, K-nearest neighbour, random forest, Support vector machine and decision tree on imbalanced as well as balanced datasets. The performance is assessed on thirteen datasets of apache projects at system and component level. System level is project as a whole whereas components are different modules with specific functionalities within a system. The results are analyzed based on following research questions:

Research Question 1: Does the performance of machine learning algorithm significantly improve on balanced dataset at system level?

Research Question 2: Does balanced datasets improves the performance of machine learning techniques at component level?

The previous works carried out by authors using class imbalance problem in various fields of software engineering has been extensively reviewed. The comparative analysis between previous work and our work is done in table 1.

The remaining paper is organized as follows: section 2 gives the related work on class imbalance in various fields. Section 3 describes the experimental design of the work followed by research methodology in section 4. Section 5 analyzes the various techniques and outcome is shown in the form of bar graphs followed by conclusion.

II. RELATED WORK

Class imbalance exists in various real-life applications where distribution of data per class is highly imbalanced. It includes video mining [19], churn prediction [20] fault prone module detection [21-22], extraction of complex relations of biomedical events [23],

Manuscript published on November 30, 2019.
* Correspondence Author
Shubhra Goyal Jindal*, University School of Information and communication technology, Guru Gobind Singh Indraprastha university, Delhi, India. Email: shubhra.phd@ipu.ac.in
Arvinder Kaur, University School of Information and communication technology, Guru Gobind Singh Indraprastha university, Delhi, India. Email: shubhra.phd@ipu.ac.in

fraud detection [24], software fault prediction [25], bioinformatics [26], detection from endoscopic videos [27] and many others.

Table-1: Comparative Analysis on bug severity and class Imbalance problem

Authors	Number of datasets (>10)	Class Imbalance problem solved	Types of dataset used
Menzies et.al., 2008	×	×	Text data
Lamkanfi et.al., 2010	×	×	Text data
Lamkanfi et.al. 2011	×	×	Text data
Chaturvedi and Singh, 2012	×	×	Text data
Malhotra et.al., 2013	×	×	Text data
Jindal et.al., 2014	×	×	Text data
Pelayo and Dick, 2007	×		Numeric data
Khoshgoftaar and Gao, 2009	×		Numeric data
Khoshgoftaar et.al., 2010	×		Numeric data
Wang and Yao, 2013	×		Numeric data
Siers and islam, 2015	×		Numeric data
Tan et.al., 2015	×		Numeric data
Chen et.al., 2016	×		Numeric data
Jiang and Zhou, 2011	×		Numeric data
Han et.al., 2016	×		Numeric data
Kaur and Goyal, 2017	×	×	Text data

In software engineering, the issue of imbalanced data is widely found and analyzed by many researchers. The author [7] focuses on defect prediction using stratification-based resampling technique. The authors use SMOTE and C4.5 decision tree classifier on four datasets of NASA project. The authors [6] learned the impact of data sampling succeeded by wrapper-based feature ranking technique for attribute selection. The results show that attribute selection becomes more potent after data sampling. In their successive study [8], two data preprocessing steps are used for defect prediction. Data sampling is done using six feature ranking techniques. A modified version of Adaboost.NC is proposed which adjust its parameters and performs better than original Adaboost.NC [9].

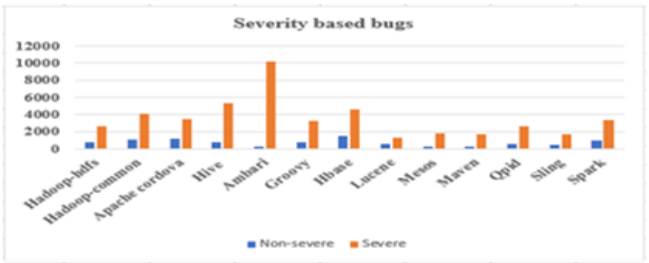
The authors propose two techniques, one cost-sensitive approach CSForest and other cost-sensitive voting technique CSVoting to inspect a solution to class imbalance problem and reduce classification cost. The techniques are evaluated on six datasets from NASA MDP repository [10]. M.tan et.al. focuses on change classification i.e modification done in a file that predicts defect at a change level using Time sensitive and online change classification. Software defect prediction improves software quality and saves time and cost in software testing [11]. A new model is proposed in which overlapping non-defective data is removed by neighborhood cleaning and balanced dataset is generated through Easy Ensemble method [12]. A consolidated and effective defect prediction model is provided for intra and inter-project [13]. Subclass discriminant analysis (SDA) is introduced improving leave-one-out-test criterion, names as ISDA. An approach is proposed to oversample the data and decrease the overfitting problem by combining SMOTE with AdaBoost algorithm on MDP dataset of NASA project to solve an imbalance problem for defect data detection [14]. Binary class imbalance problem is solved by converting into multiclassification problem for software defect detection [15]. ROCUS (Random Committee with UnderSampling) is proposed with undersampling method for imbalanced data [16]. Han et.al., proposed two online-feature selection to learn an online classifier for imbalanced data and truncated gradient method to elect subset of features to ascertain the classifier [28]. The previous research done on defect prediction using class imbalance problem is done on datasets of NASA project [6-16] and promise repository [8-9,12], which contains only numerical attributes. This study solves the imbalance problem on textual description of bugs of various Apache projects from Jira repository using SMOTE (Synthetic minority oversampling technique) data sampling technique to improve the performance of machine learning classifier.

III. EXPERIMENTAL METHOD

A. Data Collection And Statistical Tests

The proposed approach is evaluated on thirteen projects of Apache software foundation. A tool named Bug Report Collection System (BRCS) is used to extract bug reports from Jira repository [29]. The projects with highest number of bugs available are extracted. In Jira issue tracking repository, severity of bugs is categorized as blocker, critical, major, minor and trivial. In our work, they are broadly categorized as Severe (blocker, critical and major) and Non-severe (trivial and minor). The number of severe and non-severe bugs of various projects such as Apache cordova, Hadoop-hdfs, Groovy, Hadoop-common, Hive, Ambari, Hbase, Mesos, Lucene, Spark, Qpid Maven and Sling [36] are illustrate through bar graphs in Fig 1.

Fig. 1. Number of severe and non-severe bugs of various projects



Statistical test namely Friedman test is used to verify the results which are further validated through Nemenyi post-hoc test. These tests allow data to be distribution free i.e. normal distribution [30].

B. Performance Measures

Some researchers in previous studies have criticized the practice of traditional metrics such as precision, recall and accuracy while solving class imbalance problem [31-33]. They support the use of other potent metrics viz. Geometric mean (G-mean) and Balance. The accuracy of both classes, i.e severe and non-severe is augmented using Geometric mean. Confusion matrix which consists four outcomes of a classifier as shown in table II is used to compute the performance metics. The performance metrics that are used are listed in table III.

Table -II: Confusion matrix

	Correct Severity	
Predicted Severity	Non- Severe	Severe
Non-Severe(ns)	True positives(Tp)	False positives(Fp)
Severe(s)	False negative(Fn)	True negative(Tn)

Metrics used	Defined as	Mathematical Formula
G-mean	It is defined as geometric mean of positive and negative accuracy	$2\sqrt{\frac{Tp}{Tp+Fp} * \frac{Tn}{Tn+Fn}}$
Balance	It is measure of Euclidean distance between a pair of PD (probability of Detection) and PF (Probability of False alarm).	$1-\sqrt{\frac{\left(0-\left(\frac{Pf}{100}\right)\right)2+\left(1-\left(\frac{Pd}{100}\right)\right)2}{2}}$, Pf = $\frac{Fp}{Fp+Tn}\chi = \frac{-b\pm\sqrt{b^2-4ac}}{2a}$ PD= $\frac{Tp}{Tp+Fn}$

Table-III: Performance Measures

IV. RESEARCH METHODOLOGY

In this part we have explained the complete methodology of the proposed approach.

A. Data Sampling Methods

The data sampling approach concerns with altering imbalanced dataset to produce a balanced dataset. The various different forms for data sampling include undersampling i.e. removing few cases of the major class and over-sampling i.e adding some instances of minority class. The data sampling technique SMOTE (Synthetic minority oversampling technique [18] is adopted in this work.

It generates "synthetic" samples of minority class in lieu of oversampling with replacement. The approach fabricates samples in feature space instead of data space and selects its KNN randomly. Synthetic samples are produced in two steps: the difference between selected instance and its nearest neighbour is calculated and any number between 0 & 1 is selected and multiplied which is augmented to the sample chosen. Synthetic samples are linear combination of two identical samples of the minority class (M and M^R), defines as:

$$S = M + U.(M^R - M) \quad (1)$$

Where, S is the synthetic sample generated; M is chosen sample from minority class; M^R is random sample chosen among k nearest neighbour of M of minority class; U is random number ranging from 0<= U <= 1.

B. Data Pre-Processing

Data pre-processing is crucial step in text mining which includes various steps such as Tokenization, stop word removal and stemming [37]. The one-line bug descriptions are processed and document term matrix of most frequently occurring terms is formed.

C. Machine Learning Techniques

Supervised machine learning techniques viz., Naïve Bayes [34], Random forest [35], K-nearest neighbour [35], Support Vector machine [34], maximum entropy [34], and Decision tree [35] are used for prediction of severity of bug reports. The techniques are evaluated on imbalanced data as well as balanced datasets.

D. Model Development

The proposed model for severity prediction of software bugs solving class imbalance problem is depicted in Fig.2. **Various steps followed in the proposed approach are:**

- One-line description of bug reports of various projects are extracted.
- Pre-processing of textual data is performed through tokenization, stop word removal, stemming and document term matrix is created.
- Synthetic Minority Oversampling technique is applied to balance the imbalanced datasets of software bugs.
- Several machine learning techniques are applied on balanced as well as imbalanced datasets at system and component level.
- The results are evaluated using two metrics: Gmean and balance metrics which are further validated using friedman statistical tests.

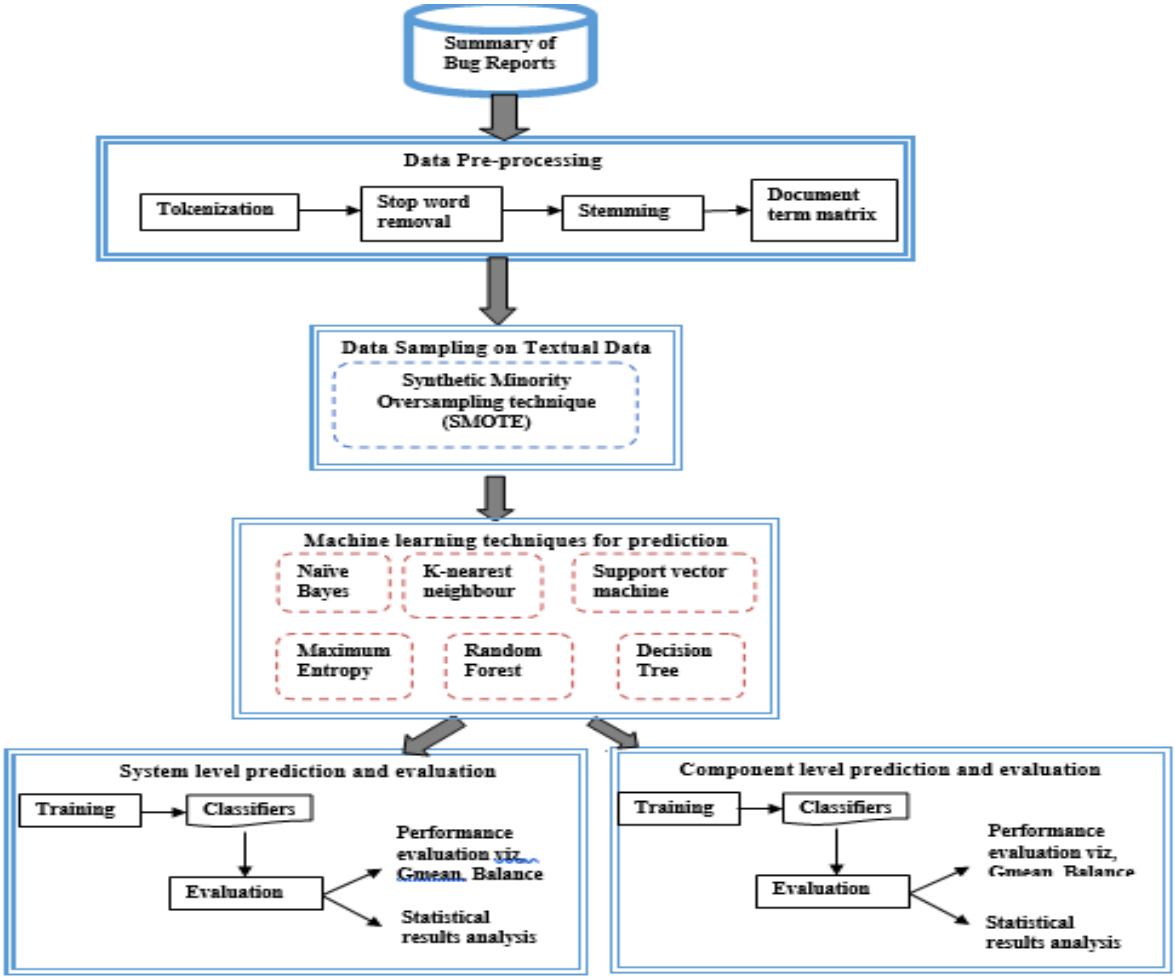


Fig.2 Framework of proposed approach

V. ANALYSIS AND RESULTS

Research Question 1: Does the performance of machine learning algorithm significantly improve on balanced dataset at system level?

Table-IV: Accuracy with and without sampling method

Project Name	NB	KNN	SVM	MAXENT	RF	DT	SMOTE
Hadoop-hdfs	75.85	75.44	76.25	76.55	77.54	77.27	75.73
Hadoop-common	77.89	75.77	79.98	79.9	80.71	80.66	79.39
Apache cordova	74.21	71.83	75.55	76.42	77.5	78.12	75.44
Hive	83.2	86.35	87.17	87.37	88.16	88.47	86.24
Ambari	96.28	95.57	97.47	97.49	97.7	97.49	97.51
Groovy	80.1	74.14	82.71	82.61	82.98	82.81	81.18
Hbase	72.7	73.24	76.71	76.9	75.54	76.54	74.89
Lucene	66.67	72.71	73.42	73.52	70.08	70.82	69.19
Mesos	84.3	84.17	87.22	87.2	87.46	86.42	87.15
Maven	81.3	83.63	85.58	86.28	86.37	85.54	84.93
Qpid	79.4	79.75	82.06	82.08	82.83	82.13	81.19
Sling	77.3	79.59	82.17	82.17	80.55	80.01	79.29
Spark	76.2	76.12	77.82	78.32	79.54	79.12	77.2

According to table IV, the accuracy of machine learning algorithms is better on imbalanced data instead of on balanced data. However, as per previous studies [31-33], accuracy is not an adequate performance assessment indicator in case of imbalanced datasets. Performance metrics such as G-mean and Balance shows decisive improvement when SMOTE is used in place of no sampling method. The Gmean metric shows an improvement of upto 34% and balance metric upto 11%. The results are illustrated through bar graphs shown in Fig. 3 and 4 with x-axis as project name, y-axis as machine learning algorithms and data sampling method and z – axis as Gmean and balance metrics respectively. The performances of various non-sampling and data sampling methods are statistically evaluated using Friedman test. Further to validate the results, post hoc Nemenyi test and Holm and Hochberg test is used.

The value of Gmean and Balance metrics of all datasets at system level are depicted in table V.

Friedman test for Non-Sampling and Sampling method using G_{mean} metric

Null Hypothesis H0: Severity prediction using distinct machine learning algorithms and class imbalance problem algorithm (SMOTE) are not significantly different when evaluated using G_{mean} metric at system level.

Alternate Hypothesis H1: Severity prediction using distinct machine learning algorithms and class imbalance problem algorithm (SMOTE) are significantly different when evaluated using G_{mean} metric at system level.

Table -V Comparison of Performance metrics with and without sampling method at system level

Project	NB	KNN	MAXENT	RF	DT	SVM	SMOTE	NB	KNN	MAXENT	RF	DT	SVM	SMOTE
	P_D							P_F						
Hadoop-hdfs	0.300	0.312	0.333	0.332	0.322	0.333	0.333	0.186	0.190	0.200	0.119	0.220	0.228	0.227
Hadoop-common	0.370	0.337	0.310	0.381	0.381	0.428	0.428	0.170	0.186	0.198	0.190	0.200	0.198	0.200
Apache cordova	0.470	0.374	0.291	0.481	0.490	0.500	0.530	0.230	0.232	0.180	0.230	0.232	0.240	0.237
Hive	0.170	0.128	0.130	0.120	0.100	0.201	0.174	0.115	0.102	0.090	0.090	0.099	0.119	0.119
Ambari	0.101	0.057	0.111	0.103	0.107	0.10	0.125	0.014	0.015	0.019	0.020	0.020	0.024	0.025
Groovy	0.331	0.218	0.251	0.401	0.410	0.33	0.413	0.150	0.160	0.169	0.169	0.170	0.174	0.172
Hbase	0.301	0.310	0.300	0.321	0.310	0.33	0.333	0.213	0.212	0.230	0.236	0.234	0.239	0.237
Lucene	0.310	0.327	0.362	0.365	0.350	0.33	0.384	0.214	0.261	0.300	0.300	0.300	0.300	0.301
Mesos	0.210	0.156	0.151	0.245	0.241	0.20	0.250	0.109	0.119	0.090	0.090	0.090	0.122	0.124
Maven	0.192	0.171	0.190	0.198	0.200	0.142	0.200	0.122	0.125	0.130	0.131	0.138	0.130	0.139
Qpid	0.220	0.209	0.223	0.230	0.170	0.25	0.235	0.146	0.130	0.131	0.143	0.139	0.176	0.177
Sling	0.300	0.208	0.322	0.302	0.320	0.285	0.352	0.188	0.188	0.183	0.159	0.159	0.190	0.195
Spark	0.121	0.140	0.161	0.162	0.101	0.166	0.104	0.177	0.104	0.210	0.210	0.211	0.402	0.212
	Gmean							*Balance*						
Hadoop-hdfs	0.042	0.081	0.220	0.220	0.100	0.063	**0.230**	0.293	0.501	0.502	0.500	0.455	0.502	0.502
Hadoop-common	0.052	0.117	0.182	0.173	0.141	0.098	**0.258**	0.538	0.513	0.492	0.569	0.570	0.572	**0.573**
Apache cordova	0.178	0.110	0.201	0.221	0.099	0.133	**0.274**	0.587	0.527	0.482	0.592	0.607	0.608	**0.628**
Hive	0.060	0.180	0.114	0.172	0.098	0.063	**0.187**	0.392	0.351	0.381	0.373	0.359	0.428	0.409
Ambari	0.103	0.083	0.106	0.108	0.111	0.112	**0.113**	0.341	0.333	0.370	0.357	0.335	0.363	**0.381**
Groovy	0.143	0.114	0.139	0.281	0.173	0.153	**0.291**	0.532	0.435	0.456	0.505	0.509	0.511	**0.567**
Hbase	0.082	0.108	0.187	0.153	0.198	0.046	**0.204**	0.445	0.409	0.419	0.433	0.481	0.497	**0.498**
Lucene	0.093	0.152	0.122	0.190	0.116	0.075	**0.168**	0.458	0.478	0.501	0.511	0.481	0.481	**0.515**
Mesos	0.106	0.112	0.114	0.115	0.115	0.115	**0.116**	0.411	0.397	0.395	0.405	0.458	0.427	**0.462**
Maven	0.144	0.132	0.150	0.100	0.099	0.111	**0.152**	0.424	0.414	0.416	0.413	0.421	0.386	**0.425**
Qpid	0.120	0.150	0.117	0.140	0.145	0.076	**0.150**	0.423	0.402	0.419	0.444	0.389	0.455	0.445
Sling	0.197	0.197	0.173	0.177	0.212	0.126	**0.212**	0.447	0.454	0.517	0.517	0.485	0.476	**0.521**
Spark	0.115	0.106	0.153	0.210	0.206	0.060	**0.228**	0.325	0.344	0.337	0.345	0.334	0.346	**0.349**

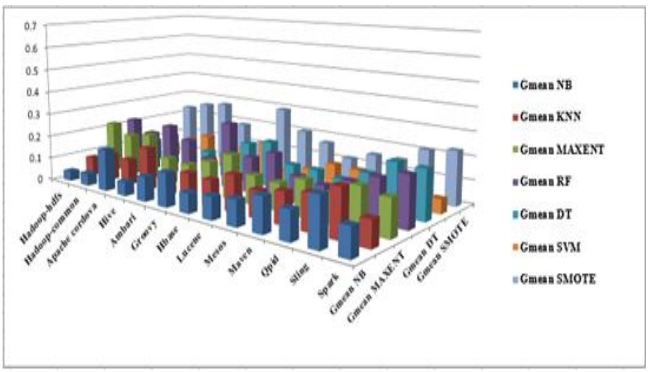


Fig 3. Gmean metric on balanced and imbalance datasets at system level

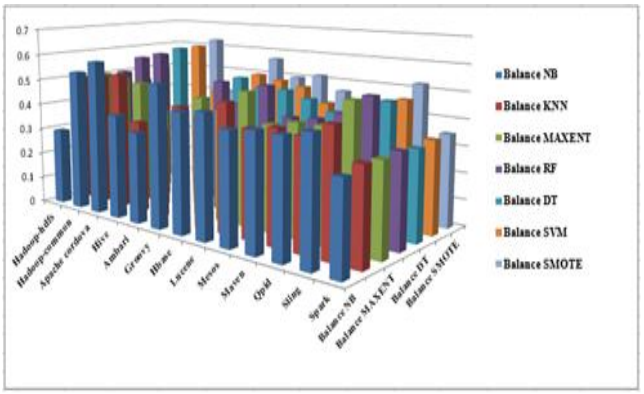


Fig 4. Balance metric on balanced and imbalance datasets at system level

It presents that out of thirteen datasets, twelve times SMOTE imbalance algorithm attains first rank. The result was significant with value of $\alpha = 0.5$ and K = 7 along with degree of freedom (df) = 72. Various algorithms achieved mean ranks of 1.115, 3.03, 3.88, 3.96, 4.03, 5.5 and 5.53 for SMOTE, RF, DT, KNN, MAXENT, NB and SVM respectively as depicted in Fig. 5. As data sampling method SMOTE significantly outperforms other Non-sampling machine learning techniques, therefore null hypothesis H0 is rejected, stating that severity prediction using distinct machine learning algorithms and class imbalance problem algorithm (SMOTE) are not significantly different when evaluated using Gmean metric at system level.

The results are further validated using Nemenyi post hoc test [29] that is used to discover significant difference among sampling method and several machine learning techniques in terms of performance. The test depicts that SMOTE sampling method is significantly different from Non-sampling methods (NB, KNN, SVM, MAXENT, RF and DT) with a critical difference (C.D) of 1.766. RF is significantly different from Naive bayes and SVM algorithms with a C.D of 1.766 and performs comparative to DT, KNN and MAXENT algorithms.

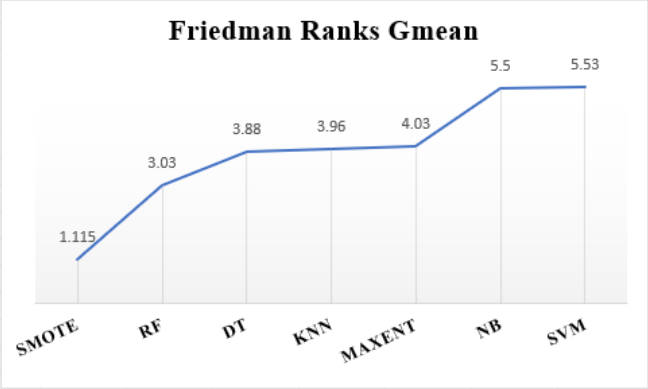


Fig.5 Friedman ranks for G_{mean} metric at system level

Friedman test for Non-Sampling and Sampling method at System level using Balance metric

Null Hypothesis H0: Severity prediction using distinct machine learning techniques and class imbalance problem algorithm (SMOTE) are not significantly different when evaluated using balance metric at system level.

Alternate Hypothesis H1: Severity prediction using distinct machine learning techniques and class imbalance problem algorithm (SMOTE) are significantly different when evaluated using balance metric at system level.

It presents that out of thirteen datasets, ten times SMOTE imabalance algorithm attains first rank and two times encounter similar rank with SVM. The result was significant with $\alpha = 0.5$, K = 7 and degree of freedom (df) = 72. Various algorithms attain mean ranks of 1.192, 2.94, 4.11, 4.423, 4.615, 4.65 and 5.92 for SMOTE, SVM, RF, DT, NB, MAXENT and KNN respectively as depicted in Fig. 6. As data sampling method SMOTE significantly outperforms other Non-sampling machine learning techniques in most of the datasets, thus null hypothesis H0 is rejected.

The results are further validated using Nemenyi post hoc test [29]. The test shows that SMOTE sampling method is significantly different from Non-sampling methods (NB, KNN, SVM, MAXENT, RF and DT) with a critical difference (C.D) of 1.766. SVM is significantly different from MAXENT, Naive nayes and KNNs algorithms with a C.D of 1.766 and performs comparative to RF and DT.

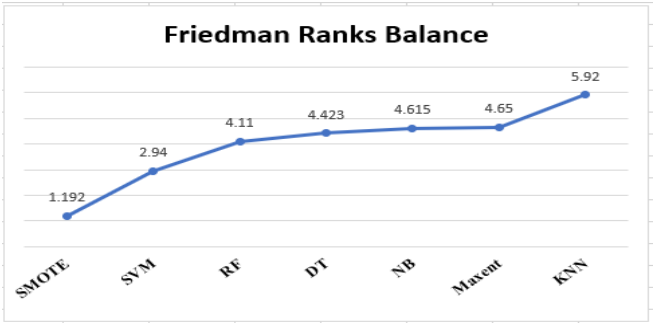


Fig.6 Friedman ranks for balance metric at system level

Class Imbalance problem using Synthetic Minority Oversampling Technique on textual description of bugs improves the performance of various machine learning algorithms to predict the severity of bugs. Performance metrics G_{mean} and Balance delivers more accurate results than accuracy metric.

Research Question 2: Does balanced datasets improves the performance of machine learning techniques at component level?

At component level, Gmean shows an increase of 42% and balance shows an increase of 62% in performance. However, in some components, some machine learning algorithm such as naïve bayes and SVM with no sampling method used was unable to identify non-severe bugs thus resulting in value zero for G-mean metric. At the same time, SVM with sampling method used encounters non-severe bugs thus resulting in a huge improvement in G-mean metric for those components. The results of component level are analyzed in the form off bar graphs as in Fig. 7 and 8 with x-axis as name of the components, y-axis as machine learning algorithms and data sampling method and z-axis as Gmean and balance metrics. The performances of various non-sampling and data sampling methods are statistically evaluated using Friedman test and further validated using Nemenyi post hoc test.

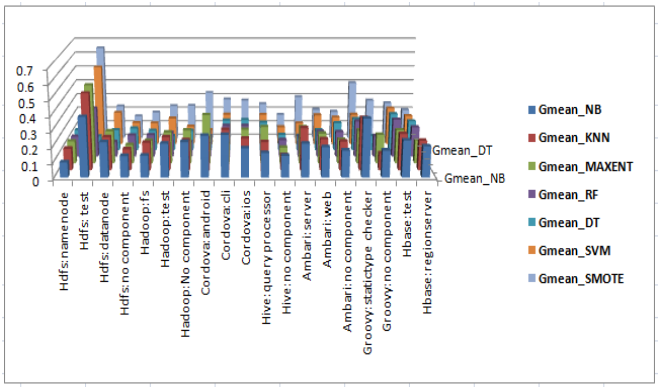


Fig 7. Gmean metric on balanced and imbalanced datasets at component level

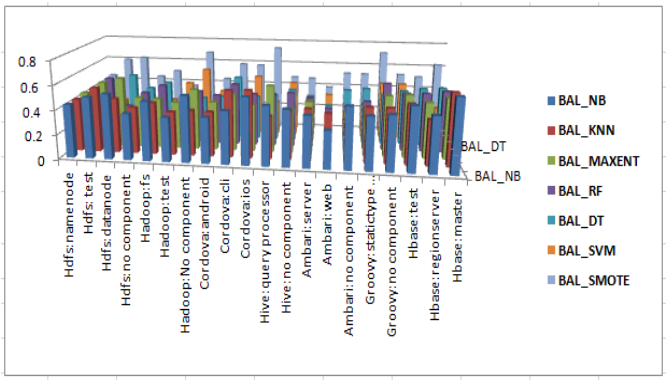


Fig 8. Balance metric on balanced and imbalanced datasets at component level

Friedman test for Non-Sampling and Sampling method at Component level using G_{mean} metric

Null Hypothesis H0: Severity prediction using distinct machine learning algorithms and class imbalance problem algorithm (SMOTE) are not significantly different when evaluated using Gmean metric at component level.

Alternate Hypothesis H1: Severity prediction using distinct machine learning algorithms and class imbalance problem algorithm (SMOTE) are significantly different when evaluated using G_{mean} metric at component level.

It presents that out of thirty-nine components, SMOTE imabalance algorithm attains first rank thirty-five times. The result was significant with $\alpha = 0.5$, $K = 7$ and degree of freedom (df) = 228. The mean ranks obtained by various algorithms are 1.16, 4.37, 4.38, 4.38, 4.42, 4.46 and 4.75 for SMOTE, SVM, RF, MAXENT, NB, KNN and DT respectively as depicted in fig.9. As data sampling method SMOTE significantly surpass other Non-sampling machine learning techniques in many components, thus null hypothesis H0 is rejected.

The results are further validated using Nemenyi post hoc test [29] to find any significant difference in the performance of sampling method and various machine learning algorithms. The result shows that SMOTE sampling method is significantly different from Non-sampling methods (NB, KNN, SVM, MAXENT, RF and DT) with a critical difference (C.D) of 1.09. Other non-sampling machine learning algorithms performs similar to each other without any significant difference in their performance.

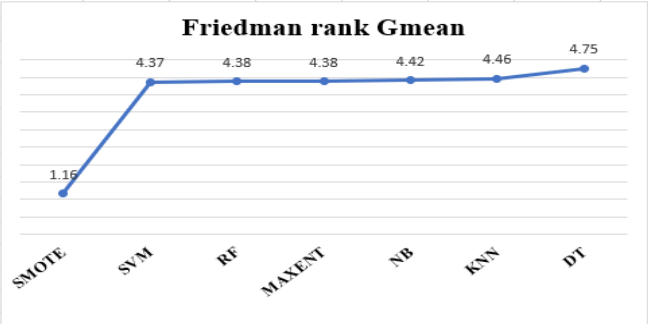


Fig. 9 Friedman ranks for G_{mean} metric at component level

Friedman test for Non-Sampling and Sampling method at Component level using Balance metric

Null Hypothesis H0: Severity prediction using distinct machine learning algorithms and class imbalance problem algorithm (SMOTE) are not significantly different when evaluated using balance metric at component level.

Alternate Hypothesis H1: Severity prediction using distinct machine learning algorithms and class imbalance problem algorithm (SMOTE) are significantly different when evaluated using balance metric at component level.

It presents that out of thirty nine components, SMOTE imabalance algorithm attains first rank thirty seven times. The result was significant with $\alpha = 0.5$, $K = 7$ and degree of freedom (df) = 228. The mean ranks obtained by various algorithms are 1.115, 3.679, 4.05, 4.217, 4.256, 4.897 and 5.5 for SMOTE, RF, KNN, NB, MAXENT , DT and SVM respectively as depicted in fig. 10. As data sampling method SMOTE significantly surpass other Non-sampling machine learning techniques in many components, thus null hypothesis H0 is rejected.

The results are validated using Nemenyi post hoc test [29] to discover any significant difference in the performance of sampling method and various machine learning algorithms. The test shows that SMOTE sampling method is significantly different from Non-sampling methods (NB, KNN, SVM, MAXENT, RF and DT) with a critical difference (C.D) of 1.09. Other non-sampling machine learning algorithms performs similar to each other without any significant difference in their performance.

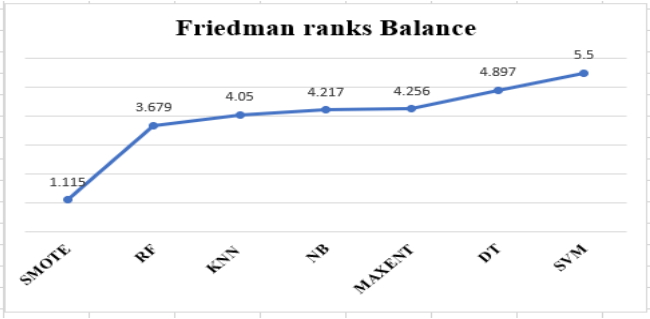


Fig. 10 Friedman ranks for Balance metric at component level

VI. CONCLUSION

Class imbalance problem had earned appreciable attention as it improves the performance of machine learning classifiers. In our research, data sampling method (Synthetic minority oversampling technique) on textual description of bugs to balance the imbalanced datasets of various projetcs of Apache Software Foundation. Various machine learning techniques namely Naïve Bayes, K-nearest neighbour, maximum entropy, random forest, support vector machine, and decision tree are applied to predict the severity of software bugs. The results of various techniques are evaluated using two potent metrics: G_{mean} and Balance on imbalanced and balanced datasets. The trend of conventional metric i.e. accuracy was checked for imbalanced datasets. It shows that performance metrics G_{mean} and balance delivers more accurate results than accuracy metric at both system and component level. Results depict that SMOTE achieves much more improved results as compared to other machine learning techniques applied on imbalanced dataset. Gmean improves by 34% and balance by 11% at system level while at component level, Gmean improves by 42% and balance by 62%.

Statistical tests, namely Friedman tests are performed to validate the results and thus, emphasizes the conclusion of our work.

REFERENCES

1. V. Engen, J. Vincent and K. Phalp, "Enhancing network based intrusion detection for imbalanced data," International Journal of Knowledge-Based and Intelligent Engineering Systems, vol. 12, no.5-6, pp. 357-367, 2008.
2. Y.H. Liu and Y.T. Chen, "Face recognition using total margin-based adaptive fuzzy support vector machines," Neural Networks, vol. 18, pp. 178-192, 2007.
3. Y.M. Huang, C.M.Hung and H.C. Jiau, "Evaluation of neural networks and data mining methods on a credit assessment task for class imbalance problem," Nonlinear Analysis: Real World Applications, vol.7, pp. 720 – 747, 2006.
4. H.L. Yin and T.Y. Leong, "A model driven approach to imbalanced data sampling in medical decision making," Stud Health Technol Inform, vol. 160, pp. 856-860, 2010.
5. Y. Jiang, J. Lin, B. Cukic and T. Menzies, "Variance analysis in software Fault prediction models," In International Symposium on Software Reliability Engineering, pp. 99-108, 2009.
6. T.M. Khoshgoftaar, P. Rebours and N. Seliya, "Software quality analysis by combining multiple projects and learners," Software quality journal, vol.17, no. 1, pp. 25-49, 2009.
7. L. Pelayo and S. Dick, "Applying novel resampling strategies to software defect prediction," In Fuzzy Information Processing Society, pp. 69-72, 2007.
8. T.M. Khoshgoftaar, K. Gao and N. Seliya, "Attribute selection and imbalanced data: Problems in software defect prediction, "In Int. Conference on Tools with Artificial Intelligence, vol. 1, pp. 137-144, 2010.
9. S. Wang and X. Yao, "Using class imbalance learning for software defect prediction," IEEE Transactions on Reliability, vol. 62, no. 2, pp. 434-443, 2013.
10. M.J. Siers and M.Z. Islam, "Software defect prediction using a cost sensitive decision forest and voting, and a potential solution to the class imbalance problem," Information Systems, vol.51, pp. 62-71, 2015.
11. M. Tan, L. Tan, S. Dara and C. Mayeux, Online defect prediction for imbalanced data," In Proceedings of the 37th International Conference on Software Engineering, vol. 2, pp: 99-108, 2015.
12. L. Chen, B.Fang, Z. Shang and Y. Tang, "Tackling class overlap and imbalance problems in software defect prediction," Software Quality Journal, pp. 1-29, 2016.
13. X.Y. Jing, F. Wu, X. Dong and B. Xu, "An improved sda based defect prediction framework for both within-project and cross-project class-imbalance problems," IEEE Transactions on Software Engineering, vol. 43, no. 4, pp. 321-339, 2017.
14. G. Li and S. Wang, "Oversampling boosting for classification of imbalanced software defect data," In 35th Chinese Control Conference, pp. 4149- 4154, 2016.
15. Z. Sun, Q. Song and X. Zhu, "Using coding-based ensemble learning to improve software defect prediction," IEEE Transactions on Systems, Man, and Cybernetics, vol. 42, no. 6, pp. 1806-1817, 2012.
16. Y. Jiang, M. li and Z.H. Zhou, "Software defect detection with Rocus," Journal of Computer Science and Technology, vol. 26, no. 2, pp. 328-342, 2011.
17. G. Goel, L. Maguire, Y. Li and S. McLoone, "Evaluation of sampling methods for learning from imbalanced data," In Int. Conference on Intelligent Computing,pp. 392-401, 2013.
18. N.V. Chawla, "Data mining for imbalanced datasets: An overview," In Data mining and knowledge discovery handbook, pp. 853-867, 2005.
19. Z.F.M. Apandi, N. Mustapha and L.S. Affendey, "Evaluating integrated weight linear method to class imbalanced learning in video data," In 3rd Conference on Data Mining and Optimization, pp. 243-247, 2011.
20. M. Bekkar, H.K. Djemaa and T.A. Alitouche, "Evaluation measures for models assessment over imbalanced data set," Journal of Information Engineering and Applications, vol. 3, no. 10, 2013.
21. Y. Kamei, A. Monden, S. Matsumoto, T. Kakimoto and K. Matsumoto, "The effects of over and under sampling on fault-prone module detection," In First International Symposium on Empirical Software Engineering and Measurement, pp. 196- 204, 2007.
22. C. Seiffert, T.M. Khoshgoftaar, J.V. Hulse and A. Folleco, "An empirical study of the classification performance of learners on imbalanced noisy software quality data," Information Sciences, vol. 259, pp. 571- 595, 2014.
23. T. Munkhdalai, O.E. Namsrai and K.H. Ryu, "H. Self-training in Significance space of support vectors for imbalanced biomedical event data," BMC bioinformatics, vol.16, no. 7, 2015.
24. C. Phua, D. Alahakoon and V. Lee, "Minority report in fraud detection: classification of skewed data," Acm sigkdd explorations newsletter, vol. 6, no.1, pp. 50-59, 2004.
25. R. Shatnawi, "Improving software fault-prediction for imbalanced data," In Int. Conference on Innovations in Information Technology, pp. 54- 59, 2012.
26. P. Yang, P.D. Yoo, J. Fernando, B.B. Zhou, Z. Zhang and A.Y. Zomaya, "Sample subset optimization techniques for imbalanced and ensemble learning problems in bioinformatics applications," Transactions on cybernetics, vol. 44, no. 3, pp. 445-455, 2014.
27. F. Deeba, S.K. Mohammed, F.M. Bui, K.A. Wahid, "Learning from imbalanced data: A comprehensive comparison of classifier performance for bleeding detection in endoscopic video," In Int. conference on Informatics, Electronics and Vision, pp. 1006-1009, 2016.
28. C. Han, Y.K. Tan, J.H. Zhu, Y. Guo, J. Chen and Q.Y Wu, "Online feature selection of class imbalance via PA algorithm," Journal of Computer Science and Technology, vol.31, no. 4, pp. 673-682, 2016.
29. A. Kaur and S. Goyal, "Bug Report Collection System (BCRS)," In Proc.of Int. Conference on Cloud computing, Data science and Engineering, 2017.
30. J. Demšar, "Statistical comparisons of classifiers over multiple data sets," Journal of Machine learning research, vol.7, pp. 1-30, 2006.
31. H. He and E.A. Garcia, "Learning from imbalanced data," IEEE Transactions on knowledge and data engineering, vol. 21, no. 9, pp. 1263- 1284, 2009.
32. T. Menzies, A. Dekhtyar, J. Distefano and J. Greenwald, "Problems with Precision: A Response to comments on data mining static code attributes to learn defect predictors," IEEE Transactions on Software Engineering, vol. 33, no. 9, 2007.
33. K. Gao, T.M. Khoshgoftaar and A. Napolitano, "Combining Feature Subset Selection and Data Sampling for Coping with Highly Imbalanced Software Data. In Proc. of Int. conference on Software Engineering and Knowledge Engineering, pp. 439-444, 2015.
34. 33. M. Costache, M. Liénou and M. Datcum, "On bayesian inference, maximum entropy and support vector machines methods," In Aip Conference Proc., vol. 872, no. 1, pp. 43-51, 2006.
35. B. Trstenjak, S. Mikac and D. Donko, "KNN with TF-IDF based Framework for Text Categorization," Int. Symp. on Intelligent Manufacturing and automation, vol. 69, pp. 1356-1364, 2014.
36. M. Mittal, R.K. Sharma and V. P. Singh, "Performance Evaluation of Threshold-Based and k-means Clustering Algorithms using Iris Dataset", Recent Patents on Engineering published by Bentham Science.
37. A. Singh, M. Mittal and N. Kapoor, "Data Processing Framework Using Apache and Spark Technologies in Big Data", In Big Data Processing Using Spark in Cloud Springer, pp.107-122, 2019.
38. A. Kaur and S.G. Jindal, "Text analytics-based severity prediction of software bugs for apache projects," International Journal of system assurance engineering and management, 2019.

AUTHORS PROFILE

Shubhra Goyal is a Research Scholar in University School of Information and Communication Technology, GGS Indraprastha University. She obtained her M.Tech in Information Technology from GGS Indraprastha University. Her research interests include Software Engineering, Object Oriented Software Engineering and data mining. She is also a student member of IEEE.

Arvinder Kaur is a Professor and dean of University School of Information & Communication Technology, GGS Indraprastha University. She obtained her Ph.D. from GGS Indraprastha University & M.E in Computer Science from Thapar Institute of Engg. & Tech. Prior to joining the school, she worked with Dr. B.R. Ambedkar Regional Engineering College, Jalandhar (1993-2000) and Thapar Institute of Engg. & Tech. (1990-1993). She has also worked for Gabriel India Ltd., Parwanoo (H.P) as Engineer (R&D). Her research interests include Software Engineering, Object-Oriented Software Engineering, Software Metrics, Microprocessors, operating systems, artificial intelligence and computer networks. She is also a lifetime member of ISTE & CSI.