Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

A Comparative Study of Rebalance Methods for Security Bug Report Prediction

Corresponding author: XXX (e-mail:).

ABSTRACT Identifying security bug report (SBR) from bug repository matters much for eliminating security issues and reducing security risk of production. However, the performance of state-of-art approaches for SBR prediction still can not meet production needs. One key reason lead to this is the class imbalance problem since the number of SBRs in production is limited. Class imbalance problem has been known as a hinder of learning performance to classification algorithms [1], [2]. Many data rebalance methods (e.g., over-sampling, under-sampling) have been proposed to deal with this challenge. Choosing an appropriate rebalance method to deal with imbalance problem is a complex task that requires to consider various criteria like effectiveness, processing speed, etc. This raises the question, which combination of rebalance method with classification algorithm can achieve the optimal performance for SBR prediction? To tackle this challenge, we perform a comparative study of 6 data rebalance methods combined with 5 popular classification algorithms for SBR prediction. To evaluate the performance of different combinations, we use 6 different sized datasets with a total of over 120k bug reports. Performance metrics Recall, Precision, F1-score, and Accuarcy are selected for the evaluation. By comparing with a state-of-art TSE work focused on SBR prediction, the results show rebalance methods could improve the performance of classification algorithms in 78% cases at least. The combination Rose (random over-sampling examples) + RF (random forest) becomes the best performing model with increasing F1-score by 104% in the best case, and 38% on average. We summarized 8 findings as guidence for choosing rebalance methods for SBR prediction.

INDEX TERMS security bug report prediction, Class imbalance, rebalance methods, classification algorithm

I. INTRODUCTION

Nowadays, security has become one of the key issues in the development of information technology [3]–[5]. As the world's most authoritative vulnerability repository CVE (i.e., Common Vulnerabilities and Exposures) reports, in 1999, there was only 894 vulnerabilities exposed; and by the end of 2018, the number has reached 110599, which is 124 times that of 1999. Figure 1 shows the vulnerability records of recent 10 years (from 2009 to 2018) exposed by CVE website. As we can see, from 2009 to 2016, around 5k vulnerabilities are recorded each year; moreover, in 2017, there's a sharp increase with 14714 records submitted, and followed with 16555 in 2018. The number of 2017 and 2018 accounts for 40% of the total of 2009 to 2018 [6]. These security issues can always cause unpredictable consequence to orgnizations and human safety. For example, the worldwide security attack WannaCry affected more than 200,000 computers across 150 countries, with total damages ranging from hundreds of millions to billions of dollars [7].

Bug reports (BRs) describe issues found during software development and maintenance [8], [9]. During the software development process, BRs are triaged with bug tracking systems (e.g., JIRA, LaunchPad, Bugzilla) and thousands of bugs are submitted each week even for a single system [**?**]. Due to the tight schedule, it's always unrealistic to fix all these bugs before each release. Previous research has show many **security-related bug reports (SBRs)** are sleeping in bug tracking system with open status until it causes severe damage in production environment [10]. Therefore, identifying SBRs as early as possible and make them fixed matters much for reducing security risks and guaranting software relibility.

Many machine learning-based approaches have been proposed for SBR prediction [4], [9]–[11]. However, the perfor-

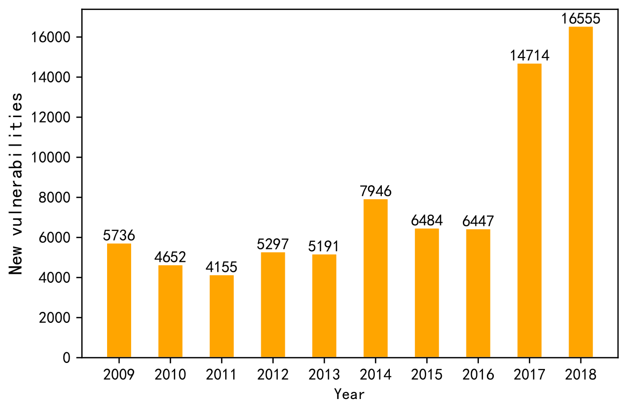

FIGURE 1: The records of vulnerabilities exposed each year from 2009 to 2018

mance of these studies still cannot satisfy industry requirements. One of the major reasons lead to this is the class imbalance problem. Class imbalance indicates when some types of data distribution significantly dominate the instance space compared to other data distributions. Given a bug report dataset B, for each $b \in B$, it might be classified depending on its **imbalance ratio (IR)**, which can be described as equation 1:

$$IR = \frac{B_{nsbr}}{B_{sbr}} \tag{1}$$

where B_{nsbr} is the class with **non-security bug report (NSBR)** in the data set, and B_{sbr} is the class of SBRs. Therefore, $IR > 1$ denotes the dataset is imbalance, the larger IR is, the heavier the class imbalance degree is for the dataset.

Class imbalance problem is a challenging and active research topic in the field of machine learning [1], [12]–[14]. One major research direction to overcome the class imbalance problem is to rebalance the original training set, either under-sampling the majority, or over-sampling the minority class. Facing the large number of approaches that have been proposed in literature, the choice of the most appropriate rebalance method itself is a complex task due to the inherent complex characteristics of imbalanced datasets: multiple criteria such as the effectiveness, the versatility of the approach with respect to classification algorithms, or the processing speed of the approach need to be considered.

In this paper, we perform a comprehensive review of existing classic imblance processing approaches by conducting a systematical experimental evaluation on 6 different sized datasets with 5 classical classifiers and 6 different data rebalance methods.

In summary, we make the following main contributions:

- We comparatively evaluate the impact of six class rebalance methods combined with different classification algorithms on 6 different-sized datasets (with a total of over 120k records) for SBR prediction.
- We summarze 8 practical findings, which would be a reference for practitioners to choose the proper rebalance methods as well as classification algorithms.
- We not only quantify the effectiveness of rebalance methods selected via several performance indicators but also demonstrate their statistical significance in terms of *Effect Size*.
- We provide our data and source code as publicly available[1].

The rest of our paper is organized as follows. Section II introduces motivation cases of this study. Section III proposes the design of our comparative analysis approach. Section IV discuss the experiment setting of our case study, while Section V presents experiment results with respect to our 3 research questions. Section VII concludes.

II. MOTIVATION

SBR prediciton can be described as a binary classification problem. The class imbalance of SBR prediction indicates the proportion of SBRs is less than that of NSBRs. However, little is known about the nature of class imbalance in SBR datasets and what strategies can be taken to tackle this challenge.

A. POOR PERFORMANCE OF MACHINE LEARNING FOR SBR PREDICTION

Many machine learning-based approaches have been proposed for SBR prediction. Table 1 summarizes the maximum performance values achieved by three representitive work [4], [10], [11] dedicated on SBR prediction with supervised-machine learning. Performance indicators *Recall* and Precision are applied in all of the three works; and the other three classical performance indicators *Accuarcy*, *F1-score* and *G-measure* are applied in one or two of them. One obvious phenomenon is that the value of same indicator varies greatly from one dataset to another. For example, the value of *Recall* range from 0.47 to 0.93 in 9 datasets of the three work; the values of *F1-score* range from 0.09 to 0.77. Even with the same approach, such as the approach proposed by Peter et al. [4], the best value for indicator *Precision* is 1 on dataset Ambari, however, the worst value 0.05 is obtained on dataset Wicket. The work proposed by Katerina et al. [11] achieves a better performance results in three NASA datasets. However, these NASA datasets as well as their source code are not publicly available.

B. THE NATURE OF IMBALANCED SBR DATASETS

There might be more than one reasons lead to the performance diversity of previous studies. However, class imbalance must be one important reason. Table 2 shows the distribution of SBRs in the datasets of these studies. As shown in this table, the proportion of SBR ranges from 0.46% to 65%,

[1]the url will be added once the paper is accepted

TABLE 1: Performance of Existing SBR prediction work

Literatures	Dataset	Recall	Precision	Accuracy	F1-score	G-measure
Michael et.al. [10]	Cissco	/	0.60	0.94	/	/
Peters et.al. [4]	Ambari	0.57	0.21	/	0.31	0.72
	Camel	0.50	0.04	/	0.08	0.54
	Derby	0.48	0.26	/	0.33	0.62
	Wicket	0.66	0.05	/	0.09	0.62
	Chromium	0.50	0.07	/	0.12	0.65
Katerina et.al. [11]	Ground Mission IV&V	0.93	0.37	0.87	0.52	0.89
	Flight Mission IV&V	0.68	0.90	0.77	0.78	0.78
	Flight Mission Developers	0.64	0.78	0.65	0.71	0.65

TABLE 2: The Distribution of Security Bug Reports

Dataset	# BRs	# SBR	% SBR	IR
Ambari	1000	20	2.00	49.00
Camel	1000	32	3.20	30.25
Derby	1000	88	8.80	10.36
Wicket	1000	10	1.00	99.00
Chromium	41940	192	0.46	217.44
Ground mission IV & V	1779	133	7.48	12.38
Flight mission IV & V	383	157	40.99	1.44
Flight mission Developers	573	374	65.27	0.53

and the IR ranges from 217.44 to 0.53 across the 8 datasets. The largest value of IR is obtained by dataset Chromium, which contains 41940 bug reports in total but only 192 SBRs; the smallest value of IR is obtained by dataset "Flight mission Developers" with 374 SBRs in 573 bug reports. On average, the IR of these datasets achieves 52.55. The IR of the three datasets applied by Katerina et al. [11] is much smaller than that of the other datasets on average.

III. METHODOLOGY

Many data rebalance methods have been introduced in the area of machine learning [1], [15]–[17]. In this paper, we conduct a comparative analysis of different rebalance methods. Figure 2 shows the framework of study. It includes three phases: (1)data preprocess; (2) training data rebalance; (3) performance evaluation and comparative analysis.

A. DATA PREPROCESS

Similar to previous studies [2], [4], [11], we use "Description" of bug reports to conduct bug report prediction as the column "Description" always contains the most detailed information of a bug report (e.g., the actual situation of the bug, steps to reproduce, expected result). We use the off-the-shelf data preprocessing techniques of text mining, such as remove stopwords (e.g., a, an ,the), text vectorization, dimensionality reduction, etc. Here we give a brief introduction of the "text vectorization" and "dimensionality reduction" approaches we used.

1) Text Vectorization

In order to feed the bug reports into the classification model, we must transform the text information bug reports to vectors. We use CountVectorizer integrated in scikit-learn [18] to tokenize the "Description" of bug reports. CountVectorizer is a kind of "bag-of-words" models. It constructs vectors by counting word frequencies in the entire training dataset. A library of words is built through training data and the number of words in the library is the vector length. A word in the vector indicates one dimension. Thus, the amount of dimensions is decided by the number of different words used to describe the bug reports.

2) Dimensionality reduction

Dimensionality reduction is widely applied for maintaining important features and removing the redundant ones of data. The purpose of dimensionality reduction is removing features with a low variance to improve estimators' accuracy scores or to boost their performance on very high-dimensional datasets [19]. In this paper we make use of *SelectFromModel* [18] coupled with *LinearSVC* [20] to evaluate feature importance and select the most relevant features for SBR prediction. *SelectFromModel* is a meta-transformer that can be used along with any estimator that has a feature importances attribute after fitting.

B. REBALANCE METHODS

To evaluate the effectiveness of different data rebalance methods on SBR prediciton, we selects 6 different data rebalance methods for our study. Four of them are classical approaches implemented in imblearn [15]: Adaptive synthetic, Borderline-SMOTE, Near Miss, and Condensed Nearest Neighbour; and the other two approaches are Rose and MAHAKIL proposed by recent work [16] and [17], respectively. We first give a brief introduction of the 4 classical approaches, and then introduce Rose and MAHAKIL in subsection III-B1 and III-B2.

- Adaptive synthetic (Adasyn) [21]: The essential idea of Adasyn is to use a weighted distribution for different minority class examples according to their level of difficulty in learning, where more synthetic data is generated for minority class examples that are harder to learn compared to those minority examples that are easier to learn.
- Borderline-SMOTE (Blsmt): Borderline-SMOTE is implemented by Han et.al. [22] on the basis of SMOTE. It only over-samples the minority examples near the borderline. There are two different versions of Borderline-SMOTE named Borderline-SMOTE1 and Borderline-SMOTE2, respectively. We use Borderline-SMOTE1 integrated in imblearn [15] in this study.

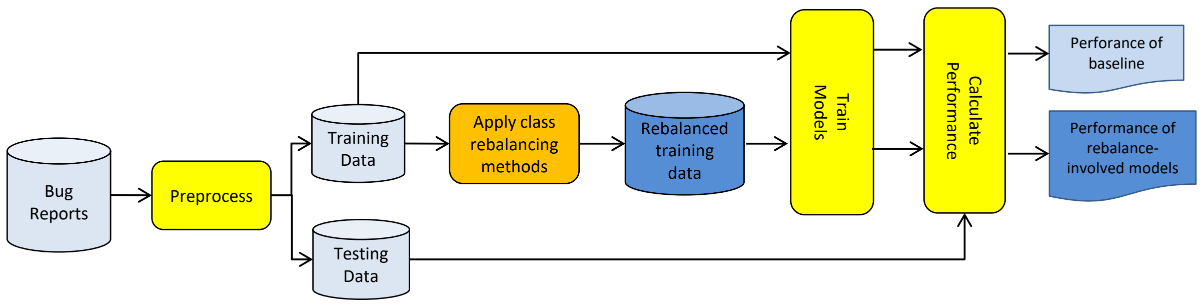

FIGURE 2: Framework of our approach

- Near Miss (NM): NM adds some heuristic rules to select samples. Near Miss heuristic rules are based on nearest neighbors algorithm and have three versions of implementation with different types of heuristic. NearMiss-3 is a 2-steps algorithm. First, for each negative sample, their M nearest-neighbors will be kept. Then, the positive samples selected are the one for which the average distance to the N nearest-neighbors is the largest. When under-sampling a specific class, NearMiss-3 is probably the version which will be less affected by noise due to the first step sample selection.
- Condensed Nearest Neighbour (CNN): uses a 1 nearest neighbor rule to iteratively decide if a sample should be removed or not. It is one of the cleaning under-sampling techniques, which do not allow to specify the number of samples to have in each class. In fact, each algorithm implements a heuristic which will clean the dataset.

1) Boostrap Random Over-Sampling Examples Technique

Boostrap Random Over-Sampling Examples Technique (Rose) [16] tends to produce more accurate and reliable estimates in the context of software engineering [23]. It provides an unified framework to deal with imbalance problem of binary classification. It combines oversampling and undersampling and uses a smoothed-bootstrapping approach to draw artificial samples from the feature space neighbourhood around the minority class. The process of Rose can be described as algorithm 1. It starts from spliting the dataset into minority class set N_{min} and majority class N_{maj} (Line 1). Then the number of the minority class set, majority class set, and the sum of both of them are calculated. (Line 2 and 3). To balance the number of majority class set and minority class set, it randomly remove items from the majority set, and copy items of the minority set to make the number of each set takes 50% of the total (Line 4 to Line 9).

2) Mahakil

Mahakil is an over-sampling approach implemented by Kwabena [17] based on chromosomal theory of inheritance. The goal of MAHAKIL is to generate a composite instance with special and common attributes. The process of MAHAKIL can be described as algorithm 2. Here, $N_{bin1} = \{y_1, y_2, ..., y_{mid}\}$, $N_{bin2} = \{y_{mid+1}, y_{mid+2}, ..., y_k\}$, $y_i \in N_{mindist}$. It consists of three steps. In the first step, after separating the minority samples from the majority samples, the separation measure value of the minority samples is calculated. This involves calculating the Mahalanobis distance (D2) for a small number of samples and arranging the samples in descending order according to D2 values. The second step is to split the sorted sample into two bins using the midpoint of the distance matrix (D2) or central instance and assign a unique label to each member of each bin in turn.

Algorithm 1: Procedure of Rose

```
1 Split dataset N into arrays of minority class $N_{min}$ and
    majority class $N_{maj}$;
2 Compute the number of samples:$T$;
3 Compute the number of minority class and majority
    class: $T_{min}$ and $T_{maj}$;
4 while $T_{maj} > \frac{T}{2}$ do
5     Remove a sample from $N_{maj}$ randomly;
      $T_{maj} = T_{maj} - 1$;
6 end
7 while $T_{min} \leq \frac{T}{2}$ do
8     Copy a sample from $N_{min}$and add to $N_{min}$ ;
      $T_{min} = T_{min} + 1$;
9 end
```

C. CLASSIFICATION ALGORITHMS

For our experiments, we use five classification algorithms which have been widely applied in the area of bug report prediction [2], [4], [10], [11]. Includes:

- Logistic Regression (LR): LR is a widely applied classification algorithm in software engineering [24]. It solves the classification problem with a similar idea to regression.
- Multinomial Naive Bayes (MNB): MNB is a popular method for text classification due to its computational efficiency and relatively good predictive performance [4], [8]. It implements the naive bayes algorithm for multinomially distributed data and calculate the conditional probability of each dimension feature. These probabilities are then combined to make a classification

Algorithm 2: Procedure of Mahakil

```
1  Split dataset N into arrays of minority class $N_{min}$ and
     majority class $N_{maj}$ ;
2  Compute Mahalanobis distance ($D^2$) for $N_{min}$,
     $D^2 = (x-\mu)^T \sum^{-1}(x-\mu)$where $\mu$=mean and
     $\sum$=covariance matrix ;
3  Save$D^2$ matrix into $N_{mindist}$ ;
4  Create $N_{bin1}$ and $N_{bin2}$ from the midpoint of
     $N_{mindist}$.;
5  foreach $y_i \in N_{bin1}$ and $y_i \in N_{bin2}$ do
6      sequentially assign unique labels ($l_i$) from
         $i = 1, 2, ..., mid$ ;
7      $X_{new}$: array for generated sample, initialized to 0 ;
8      $X_{newchk}$: keeps count of number of synthetic
         samples generated ;
9  end
10 foreach $a \in N_{bin1}and\ b \in N_{bin2}$ do
11     Generate a minority class synthetic sample $x$ by
         average(a,b) ;
12     Add $x$ to $X_{new}$ and increment $X_{newchk}$:
13     if $X_{newchk}$ < $T$ then
14         pair instances in $X_{new}$ with instances in each
             parent bin $N_{bin}$; repeat steps 8 to 11.
15     end
16     if $X_{newchk}$ < $T$ then
17         pair each set of current $X_{new}$ with it's
             immediate parent set and later with
             predecessor set of immediate parent; repeat
             steps 10 to 12 for subsequent rounds.
18     end
19     if $X_{newchk} \geq T$ then
20         assign all samples in $X_{new}$ with the label of
             the minority class (defective); add to dataset
             N.
21     end
22 end
```

prediction of the feature vector.

- Support Vector Machine (SVM): SVM is one of the best machine learning algorithms, which was proposed in 1990s and applied to many types of classification problems. It maps each sample to a point in a high-dimensional space, and then selects the points which have big impact for classification as support vectors [2]. A hyperplan is then generated as decision boundary of the two classes [25].
- Multilayer Perceptron (MLP): MLP is a feedforward neural network with one or more layers between an input layer and an output layer and trained with a back-propagation learning algorithm [26]. All parameters of the MLP are the connection weights and offsets between the layers.
- Random Forest (RF): RF [27] is a combination of tree predictors such that each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest.

IV. EXPERIMENTS SETUP

Motivated by the importance of selecting appropriate data rebalance methods for the classification problem, we will examine the impact of different rebalance methods on SBR prediction through a series of experiments guided with following three research questions:

RQ1: How does class imbalance impact the performance of classification algorithms for SBR prediction?

RQ2: Can rebalance methods improve the performance of classification algorithms for SBR prediction?

RQ3: How about the efficiency of rebalance methods?

A. BASELINE APPROACH - FARSEC

To evaluate the effectiveness of different rebalance methods, we use the out-of-the-state work proposed by Peter et al. [4] as baseline approach. We choose it for three reasons: (1) it is a state-of-art approach highly related to our work; (2) it is published on strong journal TSE; (3) it shares all their sourcecode and datasets, which makes it easy to reproduce their work.

The name of their approach is Farsec. The main idea of Farsec is to filter out noise records from NSBRs to improve SBR prediction. They first identify security related keywords and security cross words from security bug reports with feature extraction approach tf-idf (term frequency-inverse document frequency); after that, NSBRs with security related keywords are removed from the training data by ranking bug reports with bug reports score and word score, which are calculated with the 100 selected security related keywords and each keyword in feature set of a bug report. The threshold 0.75 is used for bug report score for filtering out noise bug reports.

B. DATASETS

To conduct experiments, we need projects with historical bug reports, which are labeled as SBRs or NSBRs. We totally collect 6 datasets for our study. They are: Ambari, Camel, Derby, Wicket, Chromium, and OpenStack. The first 5 of these datasets are also used by the baseline work Farsec [4]. However, while reviewing these datasets, we find there are many SBRs mis-labeled as NSBRs (this situation is also mentioned by Peter et al. [4] in the introduction part of their work). We try our best to improve the label correctness of these datasets as data correctness matters much for the effectiveness of machine learning-based approaches.

We use the automatic data labeling approach proposed by Wu et al. [28] to identify the mislabeled records of datasets Ambari, Camel, Derby, Wicket, and Chromium. The basic idea of this approach is to prepare a group of ground-truth labeled data: the number of positive data (SBR) is expected to be as many as possible; but the negative data (NSBR) required is only a small group (as suggested 50 records in their work). It then uses an iterative voting process to filter out negative items and identify positive items. The goal of using this approach in this paper is to identify the SBRs mislabeled as NSBRs. So we double reviewed the SBRs predicted by their approach by confirming with experienced developers and testers. We totally corrected 736 records of these 5 datasets, the identifies of these records are listed in Section VIII. The dataset OpenStack is constructed by our previous work [28]. It is a large-scale dataset with over 80k bug reports collect from project OpenStack [29], which is one of the most popular open-source cloud computing management systems. Table 3 presents the distribution of the 6 datasets we applied.

TABLE 3: Distribution of datasets.

Dataset	# BR	# SBR	% SBR	# NSBR	IR
Ambari	1000	34	3.40	966	28.41
Camel	1000	74	7.40	926	12.51
Derby	1000	115	11.50	885	7.70
Wicket	1000	47	4.70	953	20.28
Chromium	41940	808	1.93	41132	50.91
OpenStack	83006	332	0.40	82674	249.02
Total/Average	*128946*	*1410*	***4.89***	*127536*	***61.47***

To simulate real production situation, we order each dataset by the created date of bug reports, and then split it into two parts with 5:5. The first part (with older date) is used as training data and the second part is used for testing.

C. EVALUATION METRICS

According to *confusion matrix*, the prediction result of a BR could be four:

- TP (True Positive): an SBR is predicted as SBR;
- FP (False Positive): an NSBR is predicted as SBR;
- TN (True Negative): an NSBR is predicted as NSBR;
- FN (False Negative): SBR is predicted as NSBR.

We use Recall, Precision, F1-score, and Accuracy to measure the performance of rebalance methods combined with classification algorithms. Base on the four value of confusion matrix, the value of Recall, Precision, F1-score, and Accuracy can then be calculated as following.

- Recall, which is the percentage of correctly classified SBRs (above a minimum probability) among all verified SBRs.

$$Rec. = \frac{TP}{TP + FN} \tag{2}$$

- Precision, which is the percentage of correctly classified SBRs among SBRs and NSBRs that have been classified by the model to be SBRs (i.e., exceeding a minimum probability). Precision measures the fraction of actual SBRs in the predicted SBRs.

$$Prec. = \frac{TP}{TP + FP} \tag{3}$$

- F1-Score, which is the harmonic mean that combines both precision and recall. A higher F1-score means a better performance.

$$F1. = \frac{2 * Rec. * Prec.}{Rec. + Prec.} \tag{4}$$

- Accuracy, which is the number of correct classifications divided by the total number of classifications.

$$Acc. = \frac{TP + TN}{TP + TN + FP + FN} \tag{5}$$

Statistical Analysis: we use the Wilcoxon rank-sum test [30] to analyze the statistical significance of the combination of rebalance method and classifier. It is a non-parametric test of the null hypothesis. For the performance values of two methods compared, the null hypothesis is that there exists no significant difference between the two methods. The term significance level (α) is used to refer to a pre-chosen probability and the term *p-value* is used to indicate a probability calculated after a given study. If the *p-value* that results from Wilcoxon test is less than significance level, the difference between the two methods is identified as statistically significant. Most authors refer to statistically highly significant as $p-value < 0.001$ [8]. We further calculate *Cliff's delta* by comaring the performance values of rebalance method and Farsec. Cliff's delta quantifies the amount of difference between two non-parametric variables beyond the *p-value* interpretation [8], [31]–[33]. The magnitude and the corresponding range of Cliff's delta, which is suggested by Romano et al. [30], is shown in Table 4.

TABLE 4: Magnitude of Cliff's delta [30]

Magnitude	Range of Cliff's delta
Negligible	$\lvert d\rvert < 0.147$
Small	$0.147 \leq \lvert d\rvert < 0.33$
Medium	$0.33 \leq \lvert d\rvert < 0.474$
Large	$0.474 \leq \lvert d\rvert$

D. PARAMETER SETTING

We use the 5 classification algorithms (i.e., LR, MNB, MLP, SVM, RF) implemented in scikit-learn [18] and 4 classical rebalance methods (i.e., Adasyn, Blsmt, NM, CNN) integrated in imblearn [15]. For the two state-of-art rebalance methods Rose and Mahakil, we implement them in Python based on the detailed description of Nicola et al. [16] and Bennin et al. [17]. The default parameter values are applied for all of these algorithms.

V. RESULTS AND ANALYSIS

We first give out the baseline performance value of each indicator for all five classifiers in each dataset. Based on this, we comparatively discuss the experiment results in response to answering the three raised research questions.

A. RQ1 - THE IMPACT OF CLASS IMBALANCE FOR SBR PREDICTION.

To investigate the impact of class imbalance to the performance of classification algorithms on SBR prediction, we firstly perform each of the 5 classifiers on the 6 selected datasets with our baseline approach Farsec. The results of performance indicators Recall, Precision, F1-score and Accuracy are shown in Table 5. We take the harmonic performance indicator F1-score as main evaluation indicator for its

TABLE 5: Performance of baseline approach Farsec (*the row with best F1. is bold-faced*)

Dataset	Classifier	Rec.	Prec.	F1.	Acc.
Ambari	LR	0.30	0.04	0.07	0.84
	MNB	0.40	0.01	0.03	0.46
	MLP	0.20	0.05	0.08	0.91
	RF	**0.40**	**0.11**	**0.18**	**0.93**
	SVM	0.20	0.06	0.09	0.92
Camel	LR	0.78	0.11	0.19	0.39
	MNB	**0.65**	**0.14**	**0.24**	**0.61**
	MLP	0.70	0.12	0.20	0.50
	RF	0.91	0.10	0.18	0.22
	SVM	0.50	0.11	0.18	0.57
Derby	LR	0.56	0.18	0.27	0.63
	MNB	0.75	0.16	0.27	0.48
	MLP	0.56	0.18	0.27	0.62
	RF	0.63	0.16	0.25	0.52
	SVM	**0.41**	**0.20**	**0.27**	**0.71**
Wicket	LR	0.22	0.08	0.12	0.94
	MNB	0.22	0.04	0.06	0.88
	MLP	0.22	0.14	0.17	0.96
	RF	**0.33**	**0.15**	**0.21**	**0.95**
	SVM	0.33	0.11	0.17	0.94
Chromium	LR	0.76	0.28	0.41	0.95
	MNB	0.69	0.13	0.22	0.89
	MLP	0.73	0.26	0.38	0.95
	RF	**0.79**	**0.53**	**0.63**	**0.98**
	SVM	0.77	0.27	0.40	0.95
OpenStack	LR	0.31	0.06	0.09	0.99
	MNB	0.57	0.03	0.05	0.95
	MLP	0.38	0.05	0.08	0.98
	RF	**0.66**	**0.18**	**0.28**	**0.99**
	SVM	0.43	0.05	0.09	0.98

fairness [8], [31]. The row that gets the maximum F1-score on each dataset is highlighted in bold font.

Overall, the performance values of each indicator across the 6 datasets varies, e.g., the F1-score ranges from 0.03 (by classifier LR on Ambari) to 0.63 (by classifier RF on Chromium), the Accuracy ranges from 0.22 (by classifier RF on Camel) to 0.99 (by classifier RF and LR on OpenStack). For the 4 medium-sized datasets, the number of BRs contained in each dataset is equal (1000 each), which provides a relatively fair data foundation for the impact analysis of class imbalance. Interestingly, we find the order of the 4 datasets sorted by the maximum F1-score from largest to smallest is: Derby -> Camel -> Wicket -> Ambari, which exactly matches the order sorted by IR (the last column in Table 3) from smallest to largest. Because the larger the IR, the heavier the imbalance problem is in the dataset. Thus, this situation can somehow reflect a fact: the heavier the imbalance problem is, the smaller the performance is for supervised machine learning-based classification approaches. This conclusion also meets the results of the 2 large-sized datasets: as shown in Table 5 and Table 3, the best value of F1-score on Chromium (0.63) is much better than that of OpenStack (0.28); and the IR of Chromium (50.91) is much smaller than that of OpenStack (249.02).

Finding 1: For datasets of the same or similar size, the larger the IR, the worse the prediction performance is.

TABLE 6: Three best performing combination in terms of F1-score (*Model: Rebalance methods + Classifier*)

Dataset	Model	Rec.	Prec.	F1.	Acc.
Ambari	CNN+RF	0.40	0.17	0.24	0.95
	Mahakil+RF	0.20	0.25	0.22	0.97
	Rose+RF	0.30	0.13	0.18	0.95
Camel	Rose+RF	0.35	0.84	0.49	0.93
	CNN+RF	0.33	0.75	0.45	0.93
	Mahakil+RF	0.30	0.82	0.44	0.93
Derby	Blsmt+SVM	0.38	0.41	0.39	0.85
	Blsmt+LR	0.35	0.38	0.36	0.85
	Adasyn+MLP	0.33	0.40	0.36	0.85
Wicket	Rose+RF	0.22	0.67	0.33	0.98
	Rose+MLP	0.22	0.50	0.31	0.98
	Blsmt+RF	0.22	0.50	0.31	0.98
Chromium	Rose+RF	0.65	0.93	0.77	0.99
	CNN+RF	0.77	0.78	0.77	0.99
	Mahakil+RF	0.58	0.96	0.72	0.99
OpenStack	Mahakil+MLP	0.30	0.35	0.32	1.00
	Mahakil+RF	0.19	0.81	0.31	1.00
	Rose+RF	0.32	0.29	0.30	1.00

The best value of F1-score across the 6 datasets is obtained on dataset Chromium with value 0.63. Moreover, the maximum F1-score of the 2 large-sized datasets (Chromium and OpenStack) is better than the 4 medium-sized datasets (i.e., Ambari, Camel, Derby, Wicket) although the IR of the 2 large-sized datasets is larger. One major reason contribute to this should be the number of SBRs. As we can see from column “# SBR” of Table 3, the number of SBRs in Chromium is 808, which is the largest across the 6 datasets; and followed with OpenStack.

Finding 2: The negative impact of class imbalance problem on small-sized dataset is greater than larger-sized dataset.

B. RQ2 - EFFECTIVENESS OF REBALANCE METHODS.

To evaluate the effectiveness of rebalance methods as well as different combinations (imbalance + classifier), we first use the imbalanced strategies selected (i.e., Adasyn, Blsmt, NM, CNN, Rose, and Mahakil) to rebalance the training data. Then, we use the rebalanced training data to train the classification algorithms (i.e., LR, MNB, MLP, SVM, RF). Finally, the classification models are used to predict testing data and the value of performance indicators are calculated based on the prediction results of testing data.

Table 6 presents the results of the top-3 best performing combinations in terms of F1-score for each dataset. We now analyze the effectiveness of rebalance methods by answering 3 sub-RQs.

(1) Does the rebalance methods improve the performance of classification algorithmes for SBR prediction?

To answer this question, we compare the top-3 best performing results of rebalance methods (shown in Table 6) with the best value of Farsec (the bold-faced rows in Table 5). Let's use V_{imb} to denote the performance value of models with rebalance methods, V_{farsec} denotes the best value of baseline approach Farsec. Then, the improvement δ of rebal-

TABLE 7: Improvement of rebalance methods (*the row with maximum improvement of F1. is bold-faced*)

Dataset	Model	Rec.	Prec.	F1.	Acc.
Ambari	**CNN+RF**	**0**	**55%**	**33%**	**2%**
	Mahakil+RF	-0.5%	127%	22%	4%
	Rose+RF	-25%	18%	0	2%
Camel	**Rose+RF**	**-46%**	**500%**	**104%**	**52%**
	CNN+RF	-49%	436%	88%	52%
	Mahakil+RF	-54%	486%	83%	52%
Derby	**Blsmt+SVM**	**-7%**	**105%**	**44%**	**20%**
	Blsmt+LR	-15%	90%	33%	20%
	Adasyn+MLP	-20%	100%	33%	20%
Wicket	**Rose+RF**	**-33%**	**347%**	**57%**	**3%**
	ROSE+MLP	-33%	233%	48%	3%
	Blsmt+RF	-33%	233%	48%	3%
Chromium	**Rose+RF**	**-18%**	**75%**	**22%**	**1%**
	CNN+RF	-3%	47%	22%	1%
	Mahakil+RF	-27%	81%	14%	1%
OpenStack	**Mahakil+MLP**	**-55%**	**94%**	**14%**	**1%**
	Mahakil+RF	-71%	350%	11%	1%
	Rose+RF	-52%	61%	7%	1%
Average	–	*-30%*	*191%*	*38%*	*14%*

ance methods compared with Farsec can be calculated with Equation 6:

$$\delta = 100\% * \frac{V_{imb} - V_{farsec}}{V_{farsec}} \qquad (6)$$

Compared with Farsec, the improvement of the top-3 best performed models on each dataset are shown in Table 7. The row obtains the maximum improvement of F1-score on each dataset is highlighted with bold face. With rebalance methods, the improvement of F1-score reaches 104% (on dataset Camel) at most, and 38% on average. The Precision and Accuracy are also improved by 191% and 14% respectively on average. Although Farsec could maintain a better Recall, the Precision of it is is much smaller than models with rebalance methods (e.g., the average Precision of the best result of each dataset obtained by Farsec is 0.21, and 0.56 by models with rebalance methods).

Finding 3: Rebalance methods could improve the performance of SBR prediction. Across the 6 datasets selected, the F1-score, Precision, and Accuracy can be increased by 38%, 191% and 14% on average.

(2) How about the effectiveness of the selected rebalance methods and classifiers?

Table 6 shows the top-3 best performing models of each dataset selected based on F1-score. As can be seen from the 2nd column (Model), rebalance method Rose appears most often with 6 times distributed among 5 different datasets; followed by Mahakil with 5 times distributed among 4 different datasets. CNN and Blsmt perform moderate with 3 times each. The only one rebalance method not appears in these 18 best performing resutls is NM.

Finding 4: Among the 6 rebalance methods selected in our study, Rose performs the best on the 6 SBR datasets, followed by Mahakil; NM is the worst.

For the effectiveness of the 5 classifiers, again, go through the 2nd column of the 18 best performing results shown in Table 6, it's clear classifier RF shows 13 times distributed in 5 datasets (except Derby). Furthermore, RF also wins the best F1-score (top-1) on 67% datasets (i.e., 4 of 6: Ambari, Camel, Wicket, and Chromium). Actually, RF also wins the best F1-score in 67% datasets with baseline approach Farsec (as shown in Table 5). Classifier MLP performs best on dataset OpenStack, it also appears in the top-3 of datasets Derby and Wicket. The only one classifier not appears in the 18 best performing results is MNB.

Finding 5: Among the 5 classifiers selected in our study, RF performs the best on the 6 SBR datasets, followed by MLP, SVM, and LR; MNB performs the worst.

To construct an effective classification model for SBR prediction, we hope to identify a better performing combination of rebalance method and classifier. Out of the 18 best performed combinations (models) shown in Table 6, "Rose + RF" appears most often with 5 times in 5 datasets (except Derby). Moreover, it gets the best F1-score in 3/6 datasets (i.e., Camel, Wicket, and Chromium).

Finding 6: Across all the 30 combinations (6 rebalance methods, 5 classifiers) investigated in this paper, the combination "Rose + RF" performs the best across the 6 SBR datasets.

(3) **How about the statistical significance of rebalance methods?**

To further explore the effectiveness of rebalance methods, we evaluate the statistical significance of the top-3 best performing rebalance methods (i.e., Rose, Mahakil, and CNN) selected according to Finding 4. We calculate the Cliff's delta between the 3 selected rebalance methods and Farsec in terms of F1-score and Accuracy. To be fair, we only use classifier RF since it is outstanding in both Farsec and imbalance-involved models. We perform each of these four combinations (i.e., "Farsec + RF", "Rose + RF", "Mahakil + RF", and "CNN + RF") 50 times, and then computing the p-value and Cliff's delta by comparing the results of rebalance method-involved combination with "Farsec + RF". The results is shown in Table 8. The possitive value indicate the imbalance-involved model outperforms Farsec. According to the magnitude defination presented in Table 4, the results with magnitude "Large" is highlighted in Table 8, which indicate these approaches significantly outperforms the baseline approach Farsec. Among these 18 results, there are 14 positive values in F1-score. and 16 positive values in Accuracy. In other words, the 3 selected rebalance methods could improve the performance (e.g., F1-score, Accuracy) of classification model in 78% cases (i.e., 14/18) at least.

Finding 7: In more than 78% cases, the rebalance methods "Rose", "Mahakil", and "CNN" can significantly increase the performance of classification algorithm for SBR prediction.

TABLE 8: The Cliff's delta between rebalance methods and Farsec in terms of F1-score and Accuracy (p-value <0.05); The value with magnitude "Large" is bold-faced.

Dataset	Rebalance method	F1.	Acc.
Ambari	Rose	-0.21	-0.01
	Mahakil	-0.98	**0.82**
	CNN	0.09	0.02
Camel	Rose	**1.00**	**1.00**
	Mahakil	**1.00**	**1.00**
	CNN	**1.00**	**1.00**
Derby	Rose	0.18	**1.00**
	Mahakil	-0.80	**1.00**
	CNN	**0.91**	**1.00**
Wicket	Rose	**0.98**	**1.00**
	Mahakil	0.26	**1.00**
	CNN	**0.97**	**0.97**
Chromium	Rose	**1.00**	**1.00**
	Mahakil	**1.00**	**1.00**
	CNN	**1.00**	**1.00**
OpenStack	Rose	0.43	**0.79**
	Mahakil	**1.00**	**1.00**
	CNN	-1.00	-1.00

TABLE 9: The time cost of the 3 rebalance methods selected on the 6 datasets

Dataset	Rebalance method	Time cost(s)
Ambari	Rose	<1
	Mahakil	<1
	CNN	<1
Camel	Rose	<1
	Mahakil	<1
	CNN	<1
Derby	Rose	<1
	Mahakil	<1
	CNN	<1
Wicket	Rose	<1
	Mahakil	<1
	CNN	<1
Chromium	Rose	13
	Mahakil	20
	CNN	793
OpenStack	Rose	22
	Mahakil	104
	CNN	3201

C. RQ3 - EFFICIENCY OF REBALANCE METHODS

During the experiment of statistical analysis, we recorded the time cost of the 50 rounds on each of the dataset. We calculate the average cost of the 50 rounds for combination "Rose + RF", "Mahakil + RF", and "CNN + RF" on each dataset. Table 9 presents the results. The time cost of the 3 selected rebalance methods on the 4 small-sized datasets is less than 1 second. However, on the two large-sized datasets, the cost of the 3 rebalance methods varies. Rose is the fast of the 3. In particular, CNN performs very slow on large-sized datasets.

Finding 8: For small-sized dataset, Rose, Mahakil, and CNN all perform very fast with less than 1 second. For large-sized dataset, Rose is the most efficient strategy; and CNN is the worst.

VI. RELATED WORK

A. SECURITY BUG REPORT PREDICTION

Bug has been widely investigated from different viewpoints: security [4], [10], [11], [34], [35], performance [?], configuration [?], etc. Most of these works uses bug report *description* and/or *subject* as corpus and combines supervised machine learning with text mining for their implementation. For example, Yang et al. [2] leverage imbalanced learning strategies to identify HBRs throuth mining bug report description with classifier NBM, SVM and KNN. Peter et al. [4] propose an approach named FARSEC to improve the performance of text-based SBR identification. Goseva-Popstojanova et al. [11] apply both supervised and unsupervised classification for SBR identification again, the experiment on datasets of NASA projects shows the supervised learning outperforms the unsupervised learning, at the expense of labeling the training set.

B. IMBALANCE LEARNING

VII. CONCLUSION

In this work, we conduct a comparative evaluation of six different data rebalance methods in terms of SBR prediction, which faces greatly class imbalance problem. We use several text-mining related techniques together with classification algorithms to predict whether a bug report is SBR or NSBR with the column "Description". We perform experiments on 4 small-sized datasets ("Ambari", "Camel", "Derby", and "Wicket" with 1000 records each) and 2 large-sized dataset (Chromium with over 40k records and OpenStack with over 80k records). We compare the performance of rebalance method-involved models with baseline approach Farsec, and use classical performance indicators "Recall", "Precision", "F1-score", and "Accuracy" for evaluation. We harmonic value F1-score is taken as major performance indicator to guide the analysis of results. Based on our experiment results, we summarized 8 findings, which could be a reference for choosing data rebalance methods for SBR prediction, as well as other text-mining based classification problems.

VIII. APPENDIX

The identities of bug reports we corrected the label.

REFERENCES

[1] R. Kozik, M. Choras, and J. Keller, "Balanced efficient lifelong learning (b-ella) for cyber attack detection.," J. UCS, vol. 25, no. 1, pp. 2–15, 2019.

[2] X. L. Yang, D. Lo, X. Xia, Q. Huang, and J.-L. Sun, "High-impact bug report identification with imbalanced learning strategies," Journal of Computer Science and Technology, vol. 32, no. 1, pp. 181–198, 2017.

[3] Q. Chen, B. Lingfeng, L. Li, X. Xia, and L. Cai, "Categorizing and Predicting Invalid Vulnerabilities on Common Vulnerabilities and Exposures," in 25th Asia-Pacific Software Engineering Conference, pp. 345–354, IEEE, 2018.

[4] F. Peters, T. Tun, Y. Yu, and B. Nuseibeh, "Text filtering and ranking for security bug report prediction," IEEE Transactions on Software Engineering, vol. PP, no. 99, pp. 1–1, 2017.

[5] J. Li, B. Zhao, and C. Zhang, "Fuzzing: a survey," Cybersecurity, vol. 1, no. 1, p. 6, 2018.

[6] "CVE," 2018. https://www.cvedetails.com/vendor/11727/Openstack.html/.

[7] "WannaCry ransomware attack," 2017. https://en.wikipedia.org/wiki/WannaCry_ransomware_attack.

TABLE 10: Identities of bug reports that we corrected the labels from NSBR to SBR

Dataset	# Corrected	Identity
Ambari	14	2534, 3119, 3135, 3675, 6640
Camel	42	286, 1325, 2789, 3343, 3535, 4173, 4449, 4584, 5035, 5048, 5049, 5058, 5210, 5412, 5624, 5670, 1443, 2020, 2486, 2744, 3201, 3335, 3861, 3873, 4261, 4649, 4650, 5167, 5376, 5769, 5860, 5890, 6089, 6218, 6514, 6690, 6758
Derby	27	2480, 2543, 2713, 3316, 3336, 3497, 3571, 3732, 4148, 4414, 4432, 4764, 4779, 4895, 5277, 5280, 5291, 5336, 5412, 5493, 5552, 5736, 5951, 6006, 6212, 6273, 6440, 6553, 6565
Wicket	37	1186, 1365, 1370, 2384, 4054, 4975, 5751
Chromium	616	836, 1372, 1643, 2150, 2151, 2221, 2258, 2322, 2734, 2877, 3016, 3108, 3150, 3257, 3534, 3795, 3859, 4091, 4475, 4478, 4609, 4654, 4739, 4789, 5014, 5092, 5214, 5820, 5826, 5933, 6184, 6185, 6513, 6520, 6671, 7201, 7280, 7596, 7833, 8179, 8267, 8388, 8789, 8816, 8868, 9003, 9197, 9245, 9246, 9254, 9351, 9365, 9370, 9376, 9411, 9429, 9440, 9450, 9458, 9475, 9499, 9503, 9513, 9514, 9515, 9516, 9561, 9563, 9565, 9757, 9765, 9987, 10560, 10737, 10738, 10739, 10747, 10748, 10758, 10849, 10871, 10873, 10945, 10960, 11010, 11068, 11071, 11116, 11130, 11139, 11144, 11158, 11213, 11270, 11271, 11286, 11333, 11412, 11460, 11462, 11485, 11504, 11777, 12120, 12366, 12571, 12614, 12863, 14105, 14138, 14188, 14212, 14218, 14717, 15350, 15351, 15352, 15708, 15771, 15818, 15876, 15990, 16036, 16089, 16091, 16092, 16093, 16096, 16102, 16104, 16117, 16153, 16156, 16161, 16195, 16209, 16324, 16325, 16326, 16365, 16464, 16573, 16576, 16577, 16579, 16583, 16615, 16617, 16628, 16658, 16661, 16679, 16708, 16724, 16818, 16821, 16841, 17002, 17077, 17081, 17129, 17184, 17185, 17187, 17207, 17237, 17242, 17244, 17291, 17297, 17358, 17385, 17451, 17453, 17459, 17540, 17576, 17795, 17813, 17844, 17855, 17856, 17861, 17996, 18142, 18218, 18219, 18252, 18339, 18435, 18461, 18492, 18532, 18540, 18590, 18661, 18664, 19191, 19196, 19238, 19369, 19371, 19377, 19391, 19463, 19540, 19554, 19616, 19676, 19703, 19750, 19775, 19891, 20067, 20111, 20113, 20190, 20202, 20320, 20459, 20470, 20497, 20504, 20541, 20581, 20582, 20616, 20617, 20620, 20641, 20659, 20680, 20683, 20805, 20837, 20921, 21039, 21041, 21079, 21137, 21261, 21279, 21280, 21286, 21326, 21401, 21408, 21479, 21595, 21949, 22088, 22098, 22109, 22287, 22450, 22544, 22559, 22597, 22605, 22657, 22923, 22932, 22984, 23014, 23058, 23081, 23089, 23197, 23284, 23310, 23313, 23416, 23515, 23839, 23918, 24345, 24503, 24532, 24536, 24845, 24901, 24951, 25044, 25086, 25138, 25190, 25253, 25297, 25338, 25339, 25370, 25529, 25641, 25648, 25689, 25892, 25945, 26093, 26126, 26213, 26282, 26311, 26493, 26582, 26652, 26853, 26867, 26920, 26926, 26966, 27035, 27149, 27158, 27196, 27312, 27315, 27316, 27317, 27346, 27588, 27644, 27665, 27735, 27837, 27838, 27936, 27989, 27993, 28026, 28067, 28072, 28118, 28123, 28179, 28187, 28200, 28213, 28364, 28493, 28499, 28622, 28633, 28662, 28824, 28874, 28912, 29004, 29026, 29068, 29069, 29104, 29115, 29137, 29325, 29392, 29471, 29526, 29527, 29674, 29675, 29681, 29705, 29802, 29811, 29824, 29863, 29919, 30002, 30110, 30187, 30194, 30346, 30436, 30448, 30522, 30532, 30547, 30559, 30632, 30633, 30667, 30762, 30857, 31091, 31129, 31173, 31375, 31435, 31440, 31586, 31590, 31591, 31634, 31643, 31644, 31671, 31753, 31799, 31800, 31985, 31999, 32004, 32021, 32084, 32085, 32088, 32089, 32131, 32170, 32257, 32273, 32299, 32349, 32352, 32353, 32356, 32359, 32360, 32366, 32391, 32588, 32623, 32624, 32644, 32700, 32871, 32872, 32940, 32977, 33206, 33394, 33426, 33427, 33475, 33505, 34052, 34149, 34380, 34391, 34485, 34554, 34570, 34587, 34595, 34672, 34742, 34963, 34994, 35211, 35229, 35330, 35367, 35516, 35571, 35614, 35625, 35756, 35835, 35880, 36062, 36063, 36075, 36142, 36206, 36255, 36261, 36561, 36605, 36648, 36755, 36897, 37168, 37334, 37380, 37385, 37439, 37717, 37727, 38046, 38082, 38152, 38206, 38231, 38280, 38293, 38305, 38322, 38357, 38383, 38402, 38481, 38487, 38490, 38895, 39007, 39009, 39014, 39016, 39021, 39050, 39052, 39177, 39253, 39282, 39325, 39342, 39352, 39353, 39373, 39637, 39720, 39918, 39963, 39979, 40429, 40467, 40475, 40481, 40495, 40499, 40659, 40661, 40699, 40877, 40879, 40966, 40973, 41177, 41231, 41315, 41547, 41853, 42026, 42084, 42105, 42112, 42153, 42185, 42199, 42204, 42257, 42265, 42287, 42335, 42389, 42590, 42593, 42622, 42788, 42791, 42842, 42866, 42877, 42880, 42897, 42925, 42942, 42958, 42965, 43109, 43113, 43150, 43179, 43185, 43281, 43387, 43441, 43451, 43471, 43521, 43775, 43779, 43878, 43914, 43964, 43965, 44090, 44130, 44320, 44354, 44393, 44443, 44540, 44664, 44667, 44738, 44879, 44885, 44892, 44945, 44946, 44950, 44966, 44979, 44994, 45072, 45079, 45133, 45135, 45210, 45228, 45254, 45263, 45301, 45345, 45566, 45604, 45675, 46144, 46161, 46162, 46163, 46177, 46186, 46197, 46250

[8] Y. Fan, X. Xin, D. Lo, and A. E. Hassan, "Chaff from the wheat: Characterizing and determining valid bug reports," IEEE Transactions on Software Engineering, vol. PP, no. 99, pp. 1–1, 2018.

[9] Z. Wan, X. Xia, D. Lo, and G. C. Murphy, "How does machine learning change software development practices?," IEEE Transactions on Software Engineering, 2019.

[10] M. Gegick, P. Rotella, and T. Xie, "Identifying security bug reports via text mining: An industrial case study," in Mining software repositories (MSR), 2010 7th IEEE working conference on, pp. 11–20, IEEE, 2010.

[11] K. Goseva-Popstojanova and J. Tyo, "Identification of security related bug reports via text mining using supervised and unsupervised classification," in 2018 IEEE International Conference on Software Quality, Reliability and Security (QRS), pp. 344–355, IEEE, 2018.

[12] N. Japkowicz and S. Stephen, "The class imbalance problem: A systematic study," Intelligent data analysis, vol. 6, no. 5, pp. 429–449, 2002.

[13] C. Tantithamthavorn, A. E. Hassan, and K. Matsumoto, "The Impact of Class Rebalancing Techniques on the Performance and Interpretation of Defect Prediction Models," IEEE Transactions on Software Engineering, 2018.

[14] M. Tan, L. Tan, S. Dara, and C. Mayeux, "Online Defect Prediction for Imbalanced Data," in Proceedings - International Conference on Software Engineering, pp. 99–108, 2015.

[15] "imblearn," 2016. http://imbalanced-learn.org/en/stable/.

[16] G. M. Nicola Lunardon and N. T. Abstract, "Rose: a package for binary imbalanced learning," R Journal, vol. 6, no. 1, p. 79–89, 2014.

[17] K. E. Bennin, J. Keung, P. Phannachitta, A. Monden, and S. Mensah, "Mahakil: Diversity based oversampling approach to alleviate the class imbalance issue in software defect prediction," IEEE Transactions on Software Engineering, vol. PP, no. 99, pp. 1–1, 2018.

[18] "Scikit learn," 2009. https://scikit-learn.org/stable/index.html.

[19] H. Xie, L. Jie, and H. Xue, "A survey of dimensionality reduction techniques based on random projection," 2017.

[20] "Linearsvc in sklearn," 2011. https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html.

[21] H. He, B. Yang, E. A. Garcia, and S. Li, "Adasyn: Adaptive synthetic sampling approach for imbalanced learning," in IEEE International Joint Conference on Neural Networks, 2008.
[22] H. Han, W.-Y. Wang, and B.-H. Mao, "Borderline-smote: a new over-sampling method in imbalanced data sets learning," in International Conference on Intelligent Computing, pp. 878–887, Springer, 2005.
[23] C. Tantithamthavorn, S. McIntosh, A. E. Hassan, and K. Matsumoto, "An empirical comparison of model validation techniques for defect prediction models," IEEE Transactions on Software Engineering, vol. 43, no. 1, pp. 1–18, 2016.
[24] C. Tantithamthavorn, A. E. Hassan, and K. Matsumoto, "The impact of class rebalancing techniques on the performance and interpretation of defect prediction models," IEEE Transactions on Software Engineering, 2018.
[25] W. S. Noble, "What is a support vector machine?," Nature biotechnology, vol. 24, no. 12, p. 1565, 2006.
[26] C. M. Bishop et al., Neural networks for pattern recognition. Oxford university press, 1995.
[27] L. Breiman, "Random forests," Machine learning, vol. 45, no. 1, pp. 5–32, 2001.
[28] X. Wu, W. Zheng, X. Chen, F. Wang, and D. Mu, "Cve-assisted large-scale security bug report dataset construction method," Journal of Systems and Software, vol. 160, p. 110456, 2020.
[29] "OpenStack.Org," 2009. https://www.openstack.org.
[30] J. Romano, "Appropriate statistics for ordinal level data: Should we really be using t-test and cohen'sd for evaluating group differences on the nsse and other surveys," in Annual meeting of the Florida Association of Institutional Research., 2006.
[31] T. Yu, W. Wei, X. Han, and J. Hayes, "Conpredictor: Concurrency defect prediction in real-world applications," IEEE Transactions on Software Engineering, vol. PP, no. 99, pp. 1–1, 2018.
[32] X. Ren, Z. Xing, X. Xia, D. Lo, X. Wang, and J. Grundy, "Neural network-based detection of self-admitted technical debt: From performance to explainability," ACM Trans. Softw. Eng. Methodol., vol. 28, no. 3, p. 15, 2019.
[33] X. Chen, Y. Zhao, Q. Wang, and Z. Yuan, "Multi: Multi-objective effort-aware just-in-time software defect prediction," Information and Software Technology, vol. 93, pp. 1–13, 2018.
[34] D. Wijayasekara, M. Manic, and M. Mcqueen, "Vulnerability identification and classification via text mining bug databases," in Conference of the IEEE Industrial Electronics Society, 2015.
[35] D. Behl, S. Handa, and A. Arora, "A bug mining tool to identify and analyze security bugs using naive bayes and tf-idf," in International Conference on Optimization, 2014.