Classifying Commit Messages: A Case Study in Resampling Techniques

SeyedHamid Shekarforoush, Robert Green, and Robert Dyer
Bowling Green State University
Bowling Green, OH, USA
{sshekar, greenr, rdyer}@bgsu.edu

***Abstract*—In practice, there are a variety of real-world datasets that have an imbalanced nature where one of two classes dominates the data. These datasets are generally difficult to classify using machine learning algorithms as the skewed nature of the data has a significant impact on the training process. In order to combat this difficulty, many methods of under sampling and over sampling have been proposed in order to generate comparable data sets that are more easily classifiable. This study applies multiple resampling techniques to a set of commit messages that have been extracted from multiple Github and Sourceforge projects in order to answer the question, "Do developers discuss design?" This dataset is highly imbalanced with less than 15% of all commit messages being classified as having to do with design. Results demonstrate that the combined use of resampling as coupled with various classification algorithms yields improvements in classification over the state-of-the-art by more than 10% in terms of accuracy.**

Keywords—component; machine learning; imbalance dataset; resampling;

I. Introduction

When considering real-world data, many of the datasets encountered are highly imbalanced with one or two classes having many more instances than any other class. This leads to significant issues for machine learning algorithms, as they tend to focus on the majority class while ignoring the minority class as it has little impact on performance metrics.

One such dataset was introduced in [1] where a variety of commit messages were extracted from SourceForge and Github and classified as to whether or not each instance discussed software design. This resulted in a complete dataset containing 14% of instances that considered design and 86% of instances that did not. This paper continues the study of this dataset by seeking to improve classification results through the systematic application and evaluation of resampling techniques, with the end goal of applying these methods to a much larger dataset.

The remainder of this paper is organized as follows: Section II reviews and describes the different resampling methods and performance metrics used in this study; Section III details the proposed methodology for evaluation; Section IV presents the results of this study; and Section V concludes this study.

II. Background

This section reviews the various resampling methods applied in this study as well as various performance metrics that will be used.

A. Sampling Methods

Sampling methods generally exist in three variants: under samplers, over samplers, and hybrid samplers. All three methods maintain the goal of statistically resampling a data set in order to produce a new data set that is statistically similar while achieving improved balance between minority and majority cases. Under samplers resample the majority class in order to produce a new, smaller dataset that contains a balance between the minority and majority classes. Over samplers, on the other hand, resample the minority class in order to generate a new, generally larger dataset that contains a balance of majority and minority cases. Hybrid methods combine the two for improved results.

1) Under samplers

Random Under Sampler (RUS) is the simplest form of an under sampling method. Each example of the majority class has an equal chance of randomly being chosen and removed.

Condensed Nearest Neighbor (CNN) is based on the nearest neighbor rule with some modification. In the nearest neighbor rule, any new unclassified example is classified according to its classified closest (nearest) neighbor. This method does have some drawbacks. For instance, the method typically takes too much space for storing all samples. Accordingly, the work in [2] came up with the idea of how to shrink (condense) the sample space. The study started with two bins, store and garbage. The first sample is placed into the store and then the second sample is classified by the nearest neighbor rule using store as the reference. If the sample is classified correctly, it will be stored in the garbage bin. Otherwise, it will be placed in the store bin. This procedure repeats for the entire sample space. This process then repeats using the garbage bin as the

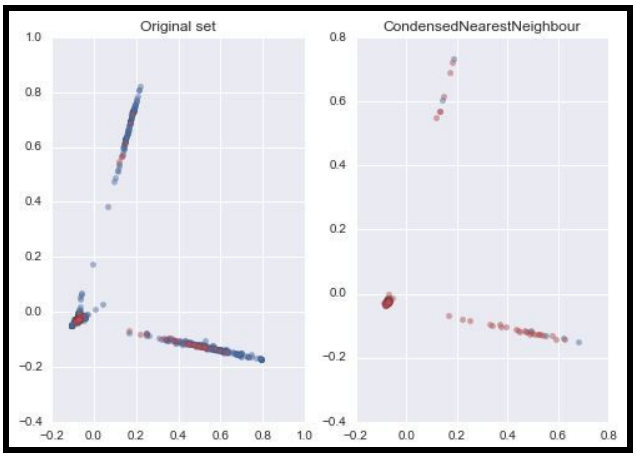

Fig. 1. CNN under sampling on the dataset used in this study

source, until the garbage is emptied or no more transfers occur. Fig. 1 shows the result of running the Condensed Nearest Neighbour under sampling method on the dataset in this study.

Tomek Links is a method based on the earlier method of condensed nearest neighbor. Tomek [3] points out some disadvantages of the condensed nearest neighbor methodology. One is that at the beginning of the condensed nearest neighbor algorithm examples are chosen randomly, making the sample spaces unnecessarily dense in the center, leading the algorithm to pay less attention to important boundary samples. This can change the boundary of the original dataset as compared to the generated and condensed nearest neighbor dataset. As a result, Tomek also proposes his own algorithm with two modifications to the condensed nearest neighbor.

Cluster Centroids (CC) is a type of under sampling method which replaces the complete majority class with the cluster centroid resulting from K-Means clustering [4].

Near Miss (NM) sampling focuses on the relation between majority and minority classes. The algorithm has three different strategies which are called Near Miss 1, 2, and 3. Near Miss 1 selects examples from the majority class that are close to three minority examples and then selects the ones with lowest average distance. Near Miss 2 is like Near Miss 1, but it also checks the distance with all minority examples and selects the ones with an average distance to the three farthest minority examples. Near Miss 3 only selects majority examples that are surrounded by minority examples [5].

One sided selection (OSS) is an under sampling method with multiple steps. First, the algorithm makes a subset of all minority examples and only one majority example. Then the one nearest neighbor algorithm is applied to this subset and the entire dataset is reclassified. The misclassified examples are then added to this subset. Secondly, Tomek Links is applied to this subset to remove the noisy and borderline examples of the majority class [6].

The fundamental idea of the Neighbourhood Cleaning Rule (NCR) is One Sided Selection, though the author claims that One Sided Selection suffers from using the nearest neighbor rule because it is too sensitive to noise in the data. To solve this problem, the author first uses the Edited Nearest Neighbor rule on the majority class to remove noisy examples. Next, examples are removed from all classes using the three nearest neighbor rule. However, this algorithm only removes examples from classes that are larger than 50% of the whole dataset [7].

The Edited Nearest Neighbours (ENN) method is based on the K nearest neighbor with a slight change [8]. The following algorithm details the Edited Nearest Neighbours method [9]:

a) Classify sample $x(i) \in D$ using k-NN with samples $x \in D$, $x \neq x(i)$;

b) Create a new design set, D', containing exactly only samples that have been classified properly from D.

Repeated Edited Nearest Neighbours (RENN) is based on the Edited Nearest Neighbors method. Tomek [9] had the idea of repeating the Edited Nearest Neighbors algorithm again and again to achieve better results, with a probability of deteriorating performance. Neither Tomek nor Wilson [8] has a concrete proof that this method will work or not. Fundamentally, in this method, Edited Nearest Neighbors is repeated infinite times (in reality the editing stops after a number of repetition due to no elimination) in order to make a new dataset. Tomek claims that, on average, this method will have better results than Edited Nearest Neighbors.

Instance Hardness Threshold (IHT) is based on the idea that each example in a dataset has a property called hardness that indicates the probability of it being misclassified. For instance, alienated or mislabeled examples have a high level of hardness. Smith and Martinez proposed an algorithm called instance hardness to measure the hardness of examples of a dataset. The data is also filtered using a constant threshold [10].

2) Over samplers

Random Over Sampler (ROS) is the naive approach to over sampling. As the name indicates, this method

simply replicates the minority class randomly until a balanced ratio with the majority class is reached.

SMOTE, or Synthetic Minority Over sampling TEchnique, is a popular over sampling method which over samples the minority class by generating synthetic examples rather than performing simple replacements. Over sampling takes one feature and its nearest neighbor, calculates the difference between the two, and then multiplies it by a random number between 0 and 1. This new sample is then added to the feature space [11]. Fig. 2 shows the result of running SMOTE over sampling method on the dataset used in this study.

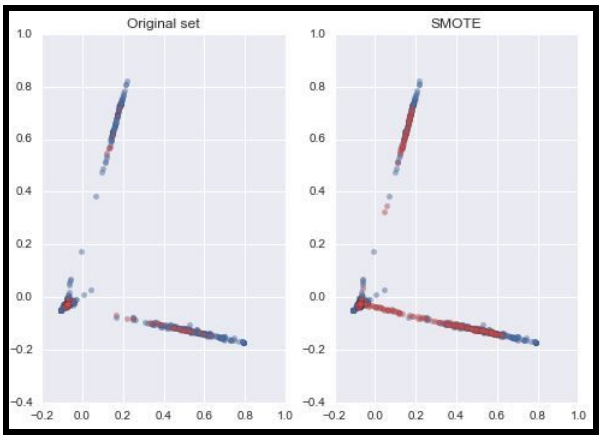

Fig. 2. SMOTE over sampling on the dataset used in this study.

SMOTE borderline 1 and borderline 2 (SMOTE B1 & B2) methods are based on SMOTE. Both claim that the borderline cases of each class are more important for classification algorithms as they are more likely to be misclassified. With this claim, these two methods try to only oversample borderline examples of the minority class as opposed to the entire minority class [12].

SMOTE SVM is another SMOTE based method that focuses on the borders between the minority and majority classes. The algorithm has two key features: First, if the original minority sample is far from majority samples, it will be resampled by extrapolation; Second, if a sample is located near the majority examples it will then be resampled to strengthen the minority class [13].

ADASYN (Adaptive Synthetic) is motivated by other synthetic sampling methods like SMOTE. ADASYN attempts to reach two goals: Reducing the bias caused by imbalanced data and shifting the boundaries toward harder examples. To accomplish this, ADASYN takes each minority example and calculates the k nearest neighbor from majority class based on euclidean distance. Next, through a weighting algorithm, ADASYN decides how many examples should be generated for each minority example. This is the key difference of ADASYN as compared to SMOTE, as an equal number of new samples are generated for each minority example [14].

3) Hybrid Methods

The SMOTE Tomek method is a combination of over sampling followed by under sampling. The authors of [15] stated that both over sampling and under sampling methods have their own disadvantages - Over sampling can cause overfitting and under sampling can eliminate useful data. Thus, a combination should improve results. Accordingly, they used a combination of both, in this case using SMOTE as an over sampling method followed by Tomek for under sampling. Fig. 3 shows the result of running the SMOTE Tomek combined sampling method on the dataset used in this study. Note the similarity in results to Fig. 2.

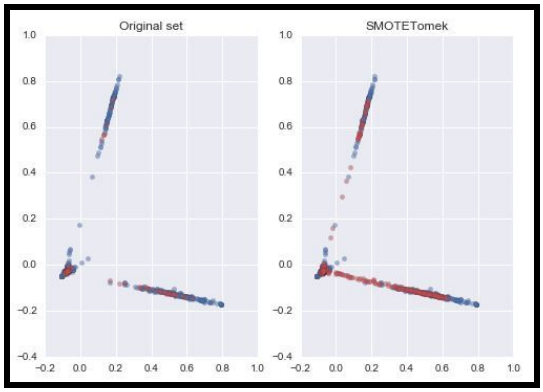

Fig. 3. SMOTE Tomek combine sampling on the dataset used in this study.

SMOTE ENN is another hybrid of over and under sampling. The idea behind this method is the same as SMOTE Tomek, an over sampling followed by an under sampling. Gustavo and Ronaldo [16] performed research and mixed different sampling methods together. This algorithm is fundamentally driven by the idea that the ENN under sampler can delete more samples than Tomek, leading to improved results.

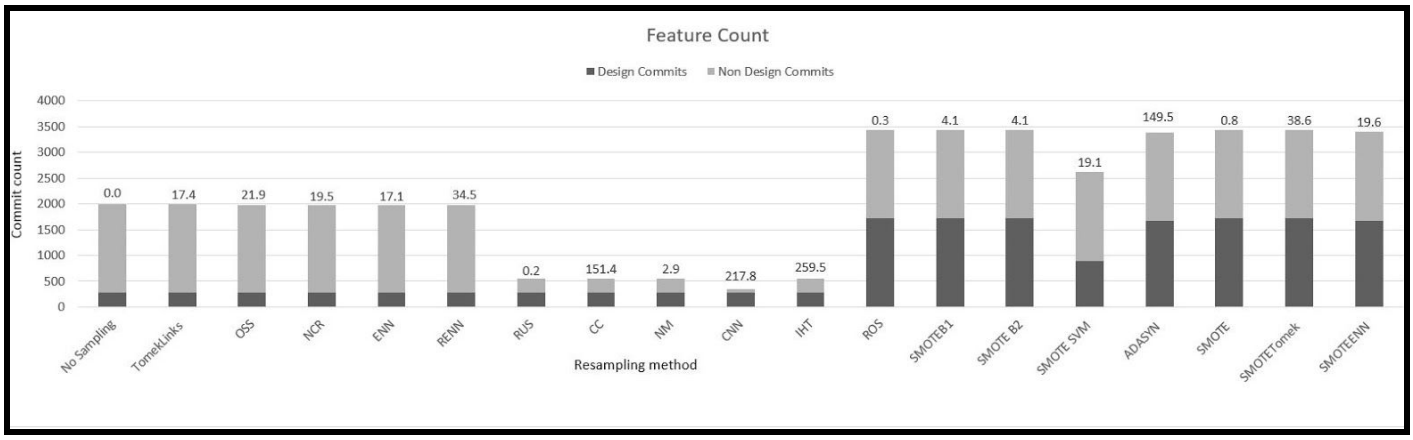

Fig. 4. Sample count of different resampling methods used in this study. The number on top of each bar represent the time it take to resample in second

B. Performance Metrics

The following metrics are used to measure the performance of various machine learning techniques. These consist of Accuracy (A), Recall (R), Specificity (S), Precision (P), F1-score (F), and G_{mean} (G) as listed in (1)-(7) where T_P is true positive, T_N is true negative, F_P is false positive, and F_N is false negative [17]. These measures provide insight into the success of any given classifier.

$$N = (T_P + T_N + F_P + F_N) \quad (1)$$

$$A = \frac{T_P + T_N}{N} \quad (2)$$

$$R = \frac{T_P}{T_P + F_N} \quad (3)$$

$$S = \frac{T_N}{T_N + F_P} \quad (4)$$

$$P = \frac{T_P}{T_P + F_P} \quad (5)$$

$$F = \frac{2PR}{P+R} \quad (6)$$

$$G_{mean} = \sqrt{RS} \quad (7)$$

The G_{mean} value is somewhat different from other measures as it makes an effort to combine multiple measures in a meaningful fashion. In all cases, a higher value is better [18].

III. Experimental setup

For this study, scikit-learn version 0.17 [19], with python version 3.5.2" was used. The scikit-learn library was used for data preprocessing, classification and evaluation. To perform resampling on the dataset, imbalanced-learn version 0.1.8 [4] was used. The overall program was run using a Windows 10 computer with Intel Core i5 6400, and 16 gigabytes of DDR4 RAM. All code and data is available online via Gitlab[1].

When data is initially read from a comma separated value (CSV) file, the default scikit-learn tf-idf (term frequency-inverse document frequency) is applied to the raw text data to convert it to a matrix of features, which is later fed to the classifiers. At this point, the feature matrix is then resampled using imbalanced-learn resampling methods. All the resampling methods in this research are applied with the default settings of the imbalanced-learn library. Having all the resampled data now in place, it is fed into classification algorithms using the default settings of the classifiers in the scikit-learn library.

For each of the different types of resampling, multiple types of classifiers are also used. These include Random Forest classifier (RF) [20], Decision Tree (DT) [21], Support Vector Classification (SVC) [22], Linear Support Vector Classification (LSVC) [23], Bernoulli Naive Bayes (BNB) [24], Nearest Centroid (NC) [25], and Multinomial Naive Bayes (MNB) [26].

A. Experiment 1

we have done different experiments using different datasets. in our first dataset we combine all of the data from both Github and Sourceforge to form a single dataset. and then train and test the dataset on itself.

In order to ensure consistent results, 10 fold cross-validation is used. This means that training and testing both happens on the resampled data, 9 fold for training and 1 fold for testing. At this time, prediction accuracy and other metrics are calculated using default modules in the scikit-learn library. Results reported are average values.

B. Experiment 2

In our second and third setting we use each of the datasets (Github and Sourceforge) and train the classifiers on resampled version of them, then test it on the other.

C. Experiment 3

later on we change and tweak some of settings of our feature extraction to see any changes in the results. We change the ngram setting of TF-IDF from (1,3) to (1,1) . And run another experiment with countVectorizer instead of TF-IDF. The results of these changes were not significant, so we will

[1] https://gitlab.com/ensemble-classifiers/fourd-python

not include them in the result section. we did these experiment on dataset setting from experiment 2.

IV. Results

Results for this study will be compared in terms of samples generated by each resampling method, the required computation time for each resampling method, accuracy, precision, recall, F1-score, and G_{mean}.

A. Sample counts and time

The first result to discuss is the number of features that each resampling method produces (i.e. the size of the resampled data set). Fig. 4 shows the number of samples in each class (both minority and majority) after applying a given resampling method. Above each bar in the figure there is also a number that shows the time it takes to resample the dataset in seconds. All names of resampling methods in the figures are abbreviated for better readability. As expected, both random methods take the lowest amount of time to create a new dataset. Apart from these two methods, the lowest computation time in each category belongs to NM and SMOTE. Another expected result, as indicated by their names, is that RENN takes twice the time to resample than ENN. IHT also takes the longest time in under sampling along with ADASYN in the over sampling category. Also of note is that Tomek Links, though an under sampling method, only slightly reduces the size of the dataset.

TABLE I. ACCURACY OF DIFFERENT CLASSIFICATION METHODS USING DIFFERENT RESAMPLING (MIN AND MAX VALUES IN BOLD).

	RF	**DT**	**SVC**	**LSVC**	**BNB**	**NC**	**MNB**
No Sampling	0.84	0.81	0.86	0.85	0.84	0.86	0.66
Tomek Links	0.84	0.81	0.86	0.85	0.84	0.86	0.65
OSS	0.84	0.81	0.86	0.85	0.84	0.86	0.67
NCR	0.84	0.81	0.86	0.85	0.84	0.86	0.65
ENN	0.84	0.81	0.86	0.85	0.84	0.86	0.66
RENN	0.84	0.81	0.86	0.85	0.84	0.86	0.66
RUS	0.59	0.63	0.57	0.68	0.6	0.72	0.69
CC	0.61	0.62	0.67	0.69	0.61	NA	0.67
NM	0.74	0.76	0.88	0.86	0.87	0.89	0.88
CNN	0.67	0.63	0.8	0.75	0.8	0.77	0.64
IHT	0.7	0.69	0.63	0.79	0.65	0.83	0.68
ROS	0.95	0.87	0.52	0.95	0.84	0.82	0.72
SMOTE B1	0.9	0.85	0.78	0.91	0.87	0.84	0.86
SMOTE B2	0.9	0.85	0.6	0.9	0.83	0.84	0.89
SMOTE SVM	0.88	0.82	0.66	0.88	0.83	0.84	0.85
ADASYN	0.86	0.8	**0.51**	**0.97**	0.75	NA	0.7
SMOTE	0.94	0.87	0.63	0.95	0.88	0.83	0.76
SMOTE Tomek	0.94	0.88	0.59	0.95	0.88	0.83	0.76
SMOTE ENN	0.94	0.88	0.51	0.96	0.88	0.83	0.76

B. Accuracy

The fundamental goal of applying resampling to a dataset is the hope that it will help improve the results during classification. Table I lists the resulting classification accuracy of various combinations of resampling and classifiers. The lowest accuracy scores are from random under sampling and Cluster Centroid methods. The highest accuracy belongs to ADASYN while using Linear SVC classifier. Surprisingly, the Random oversampler has a very high accuracy. The SVC classifier also has better accuracy while using the under sampling dataset.

Looking at Accuracy in conjunction with the data in Fig. 4, it can be seen that RUS, CC, NM, CNN and IHT resampling methods severely reduce the number of samples of the majority class when compared to other under sampling methods. This behavior has an effect on the accuracy, as it can easily be seen that these methods score lower in accuracy as compared to other under samplers.

However, accuracy alone does not guarantee the best results as sometimes the accuracy can be misleading. The accuracy paradox states that some prediction models with lower accuracy can perform better, therefore other metrics should also be considered.

C. Precision

Precision, or positive predictive value, is a measure of classification exactness that refers to how well a certain result is reproducible given the same starting conditions. Table II shows results that can be interpreted in light of the accuracy. Starting with the lowest precision, no sampling, Tomek links, OSS, NCR, ENN and RENN all had a similar and very low precision score when coupled with any classifier, yet their accuracy ratings were solid. The MNB method achieved the lowest precision score possible of zero. It appears that despite having a solid performance in terms of accuracy, these resampling methods result in low precision when coupled with classifiers.

Another group of resampling methods discuss in accuracy section were RUS, CC, NM, CNN and IHT. All of these methods had a lower accuracy than those previously discussed (Tomek links, OSS, NCR, ENN, and RENN), yet all have a much higher precision score, suggesting that, though they resulted in lower accuracy when coupled with various classification methods, their higher precision rating makes them more attractive for use. This increase in precision was due to an increased value of T_N in all cases.

Considering over-samplers and hybrid samplers, accuracy has an inverse relationship with their precision score. As expected, the coupling of various resampling methods and classifiers resulted in different pairings of accuracy and precision. For example, when coupled with classification methods the ADASYN with LSVC methods show superior accuracy with a score of 97%, and a 94% precision. The highest precision score achieved was when using the SMOTE

B1 & B2 method with SVC (99%), which also resulted in a slightly lower accuracy (78% & 60%).

In general, all of the over sampling and hybrid sampling methods have a better precision score than the under samplers, such that their average precision score of 80% is superior to the under samplers average precision score 48%.

D. Recall

Recall, or sensitivity, is a measure of classifier completeness. Recall can be a very important in some problems as it is related directly to the number of false negatives. As in the previous section, analysis shows that one particular grouping of resampling methods - no sampling, Tomek links, OSS, NCR, ENN and RENN - tend to have similar (and low) recall scores despite resulting in a high accuracy. This is explained by the fact that the number of T_N was less than F_N in these case, with the NC classifier being the exception.The group consisting of RUS, CC, NM, CNN and IHT has higher recall score than the previous group discussed, despite their lower accuracy. Again, this suggests that this grouping of resampling methods provides superior results. ROS also has a surprisingly high recall score that was also consistent across all runs.

Again, it can generally be stated that the the over resampling and hybrid resampling methods result in a better recall score than the under sampling method.

TABLE II. PRECISION (P), RECALL (R) AND SPECIFICITY (S) OF DIFFERENT CLASSIFICATION METHODS USING DIFFERENT RESAMPLERS.

	RFC			**DTC**			**SVC**			**LSVC**			**BNB**			**NC**			**MNB**		
	P	***R***	***S***	***P***	***R***	***S***	***P***	***R***	***S***	***P***	***R***	***S***	***P***	***R***	***S***	***P***	***R***	***S***	***P***	***R***	***S***
No Sampling	.23	.05	.97	.30	.29	.89	.00	.00	1.00	.33	.10	.97	.35	.15	.95	.24	.66	.66	.00	.00	1.00
Tomek Links	.19	.03	.98	.31	.30	.89	.00	.00	1.00	.37	.10	.97	.36	.15	.95	.24	.67	.65	.00	.00	1.00
OSS	.14	.03	.97	.30	.30	.89	.00	.00	1.00	.37	.10	.97	.36	.15	.95	.24	.67	.65	.00	.00	1.00
NCR	.25	.05	.98	.32	.29	.90	.00	.00	1.00	.38	.10	.97	.36	.15	.95	.24	.67	.65	.00	.00	1.00
ENN	.21	.04	.97	.31	.29	.89	.00	.00	1.00	.38	.10	.97	.36	.15	.95	.24	.67	.65	.00	.00	1.00
RENN	.26	.05	.98	.31	.30	.89	.00	.00	1.00	.38	.10	.97	.36	.15	.95	.24	.67	.65	.00	.00	1.00
RUS	.76	.32	.90	.68	.50	.77	.54	.90	.24	.73	.60	.78	.81	.32	.93	.62	.78	.53	.68	.66	.69
CC	.75	.33	.89	.64	.51	.72	.67	.71	.64	.74	.58	.80	.56	1.00	.21	.73	.55	.79	NA	NA	NA
NM	.81	.58	.86	.85	.64	.89	.86	.91	.85	.87	.85	.87	.86	.89	.85	.86	.90	.85	.94	.84	.94
CNN	.82	.71	.47	.79	.67	.40	.77	1.00	.00	.76	.92	.01	.77	1.00	.00	.75	.79	.12	.77	1.00	.00
IHT	.84	.56	.90	.72	.73	.71	.57	.98	.28	.79	.87	.76	.61	.88	.43	.60	.90	.40	.87	.82	.87
ROS	.91	1.00	.90	.81	1.00	.76	.52	.91	.15	.91	1.00	.91	.83	.85	.83	.68	.88	.58	.75	.96	.69
SMOTE B1	.96	.85	.97	.85	.87	.84	.99	.54	.99	.94	.87	.95	.91	.82	.92	.94	.77	.95	.82	.88	.80
SMOTE B2	.96	.85	.96	.85	.86	.84	.99	.37	1.00	.94	.86	.94	.88	.76	.90	.98	.78	.99	.82	.88	.81
SMOTE SVM	.92	.72	.97	.76	.76	.87	.00	.00	1.00	.89	.76	.95	.81	.70	.91	.86	.70	.94	.79	.74	.90
ADASYN	.89	.79	.91	.77	.86	.76	.00	.00	1.00	.94	.99	.94	.79	.66	.83	.76	.58	.82	NA	NA	NA
SMOTE	.93	.94	.93	.83	.93	.81	.59	.87	.40	.92	1.00	.91	.90	.86	.91	.72	.85	.67	.76	.98	.68
SMOTE Tomek	.92	.95	.92	.83	.93	.81	.56	.56	.60	.92	1.00	.91	.90	.86	.91	.72	.85	.66	.76	.97	.69
SMOTE ENN	.93	.95	.93	.83	.94	.81	.00	.00	1.00	.92	1.00	.91	.90	.86	.91	.71	.85	.67	.76	.97	.69

E. F1-score

F1-score (or F score or F measure), is a weighted average of precision and recall. In other words, this measure is the harmonic mean of precision and recall. There are other ways to average precision and recall that include the arithmetic or geometric mean, but since we are working with percentages (R and S), it more suitable to use the F1-score.

To fully understand the F1-score, consider an example: In case 1 there exists 50% precision and 20% recall. In case 2 there is 40% precision and 30% recall. The F1-score of case 1 is 28% and case 2 is 34%. Even though the arithmetic mean is the same, the F1-score of case 2 is better, therefore it is judged as a better classifier. This suggests that, the F1-score can provide an additional level of insight on the tradeoffs of various classifiers. All results for F1-score are listed in Table III. In all cases, the results of the F1-score are very similar to those of both precision and recall.

F. G_{Mean}

G_{Mean} or geometric mean is a single value representation of the confusion matrix. It is a measure of the quality of classification. There are also other representations of confusion matrices available, like Matthews correlation coefficient (MCC), though evaluating these is left for future work..

Once again, the general results of G_{Mean} (shown in Table III) are very close to the F1-score, precision, and recall. However, when comparing the G_{Mean} with F1-score we see that over samplers have a small difference with few exceptions. On the other hand, the G_{Mean} and F1-score of under samplers has a

considerable difference. The CNN under sampler has the largest differences on all classifiers.

TABLE III. F1-SCORE (F) AND G_{MEAN} (G) OF DIFFERENT CLASSIFICATION METHODS USING DIFFERENT RESAMPLING

	RFC		DTC		SVC		LSVC		BNB		NC		MNB	
	F1	***G***	***F1***	***G***	***F1***	***G***	***F1***	***G***	***F1***	***G***	***F1***	***G***	***F1***	***G***
No Sampling	.09	.23	.30	.51	.00	.00	.15	.31	.21	.38	.35	.66	.00	.00
Tomek Links	.05	.18	.31	.52	.00	.00	.16	.32	.22	.38	.35	.66	.00	.00
OSS	.05	.16	.30	.51	.00	.00	.16	.32	.22	.38	.35	.66	.00	.00
NCR	.08	.21	.30	.51	.00	.00	.16	.32	.22	.38	.35	.66	.00	.00
ENN	.07	.20	.30	.51	.00	.00	.16	.32	.22	.38	.35	.66	.00	.00
RENN	.08	.22	.31	.52	.00	.00	.16	.32	.22	.38	.35	.66	.00	.00
RUS	.45	.54	.58	.62	.68	.46	.66	.68	.45	.54	.69	.64	.67	.68
CC	.46	.55	.57	.60	.69	.68	.65	.68	.72	.46	.62	.66	NA	NA
NM	.67	.71	.73	.76	.89	.88	.86	.86	.88	.87	.88	.88	.88	.89
CNN	.76	.58	.72	.52	.87	.00	.83	.09	.87	.00	.77	.31	.87	.00
IHT	.68	.71	.72	.72	.72	.52	.82	.81	.72	.62	.72	.60	.84	.85
ROS	.95	.95	.89	.87	.66	.37	.96	.95	.84	.84	.77	.71	.84	.81
SMOTE B1	.90	.90	.86	.85	.70	.74	.90	.90	.86	.86	.85	.86	.85	.84
SMOTE B2	.90	.90	.86	.85	.49	.61	.90	.90	.82	.83	.87	.88	.85	.84
SMOTE SVM	.81	.84	.76	.81	.00	.00	.82	.85	.75	.80	.77	.81	.77	.82
ADASYN	.84	.85	.81	.80	.00	.00	.97	.97	.72	.74	.66	.69	NA	NA
SMOTE	.94	.94	.88	.87	.70	.59	.96	.95	.88	.88	.78	.75	.85	.82
SMOTE Tomek	.94	.93	.88	.87	.48	.58	.96	.95	.88	.88	.78	.75	.85	.82
SMOTE ENN	.94	.94	.88	.87	.00	.00	.96	.95	.88	.88	.78	.75	.85	.82

G. Experiment 2

in this section we are going to present the results from our second experiment. Tables IV,V and VI are the results from the second experiment. As shown in these tables none of the increased measurement from experiment 1 achiched here. In most cases we even see a drop in results compare to the un-resampled dataset, and in best cases we achieve similar results similar to un-resampled dataset.

V. Conclusion

This study has applied a variety of resampling methods (under, over, and hybrid) in combination with various classifiers in order to improve the classification of commit messages in software repositories. Particularly, this study focuses on a dataset concerned with the fundamental question, “Do developers discuss design?”. The proposed methodology improves the state-of-the-art in multiple categories by achieving an accuracy of 97% using LSVC with ADASYN, precision of 99% using SVC with SMOTE B1 & B2, recall of 100% in 10 cases, 4 using LSVC classifier and 3 using ROS over sampler, a F1-score of 97% using LSVC with ADASYN, and a G_{Mean} score of 97% using LSVC with ADASYN. All things considered, the overall best performing combination of resampling and classification occurred when using ADASYN over sampling with LSVC, resulting in an accuracy of 97%, F1-measure of 97%, and G_{Mean} of 97%. However, if computational time were a factor, the best choice would be the Random over sampler as used with the random forest classifier (A=95%, F1=95% and MCC=90%).

Analysing the second experiment shows that resampling doesn't always give us better results. we need to have large enough unbalanced starting dataset.

VI. Future works

In the future, this work may be extended by:

1. Improving parameter selection of resampling and classification methods. Many of the resampling methods and classifiers have various parameters that can be adjusted to improve performance. This may be

done through various searching techniques including grid search and population-based metaheuristics.

2. Exploring the application of ensemble-based methods like Easy Ensemble and Balance Cascade [27].
3. Evaluating the dataset at different levels. In previous work, commit messages were divided based on their source - SourceForge or Github. As this work considered only the combined dataset, a more thorough evaluation would be of interest.
4. Expanding the dataset to a larger sample size. As the current dataset contains only 2,000 samples, expanding to a much larger size would be of great interest.
5. A more in depth analysis of the resamplers, and the interdependencies of different resamplers with classifiers. Do all classifiers need resampling? and also the limitation of resampling due to the dataset size or classifiers.
6. Using natural language instead of TF-IDF for our feature extraction. Since the commits are written by human developers.

	RF	**DT**	**SVC**	**LSVC**	**BNB**	**NC**
No Sampling	0.85	0.80	0.86	0.86	0.85	0.26
Tomek Links	0.85	0.80	0.86	0.86	0.85	0.25
OSS	0.85	0.80	0.86	0.86	0.85	0.34
NCR	0.86	0.80	0.86	0.86	0.85	0.25
ENN	0.86	0.80	0.86	0.86	0.85	0.25
RENN	0.85	0.80	0.86	0.86	0.85	0.25
RUS	0.86	0.69	0.14	0.40	0.59	0.17
CC	0.85	0.83	0.16	0.60	0.14	0.39
NM	0.86	0.85	0.14	0.14	0.14	0.14
CNN	0.86	0.82	0.86	0.81	0.86	0.85
IHT	0.85	0.42	0.14	0.16	0.14	0.14
ROS	0.85	0.82	0.14	0.85	0.77	0.37
SMOTE B1	0.86	0.81	0.14	0.85	0.86	0.64
Smote B2	0.86	0.81	0.14	0.85	0.85	0.55
SMOTE SVM	0.85	0.81	0.28	0.84	0.85	0.21
ADASYN	0.86	0.80	0.86	0.86	0.85	0.50
SMOTE	0.85	0.81	0.86	0.86	0.78	0.86
SMOTE Tomek	0.85	0.82	0.14	0.85	0.86	0.70
SMOTE ENN	0.86	0.81	0.86	0.85	0.86	0.78

TABLE IV. EXPERIMENT 2 ACCURACY OF DIFFERENT CLASSIFICATION METHODS USING DIFFERENT RESAMPLING

TABLE V. EXPERIMENT 2 PRECISION (P), RECALL (R) AND SPECIFICITY (S) OF DIFFERENT CLASSIFICATION METHODS USING DIFFERENT RESAMPLERS

	RFC			**DTC**			**SVC**			**LSVC**			**BNB**			**NC**		
	P	***R***	***S***	***P***	***R***	***S***	***P***	***R***	***S***	***P***	***R***	***S***	***P***	***R***	***S***	***P***	***R***	***S***
No Sampling	0.05	0.00	0.99	0.14	0.08	0.92	0.00	0.00	1.00	0.00	0.00	1.00	0.43	0.15	0.97	0.13	0.79	0.17
Tomek Links	0.12	0.01	0.99	0.13	0.07	0.92	0.00	0.00	1.00	0.00	0.00	1.00	0.43	0.15	0.97	0.14	0.84	0.16
OSS	0.13	0.00	0.99	0.13	0.07	0.92	0.00	0.00	1.00	0.00	0.00	1.00	0.43	0.15	0.97	0.14	0.72	0.28
NCR	0.10	0.00	0.99	0.15	0.09	0.92	0.00	0.00	1.00	0.00	0.00	1.00	0.43	0.15	0.97	0.14	0.86	0.14
ENN	0.37	0.01	1.00	0.13	0.08	0.92	0.00	0.00	1.00	0.00	0.00	1.00	0.43	0.15	0.97	0.14	0.86	0.15
RENN	0.07	0.00	0.99	0.14	0.08	0.92	0.00	0.00	1.00	0.00	0.00	1.00	0.43	0.15	0.97	0.14	0.86	0.15
RUS	0.40	0.02	0.99	0.16	0.25	0.76	0.14	1.00	0.00	0.14	0.63	0.36	0.26	0.41	0.62	0.14	0.96	0.04
CC	0.31	0.01	0.99	0.18	0.06	0.95	0.14	0.96	0.03	0.15	0.41	0.63	0.14	1.00	0.00	0.15	0.65	0.35
NM	0.23	0.00	1.00	0.27	0.05	0.98	0.14	1.00	0.00	0.14	1.00	0.00	0.14	1.00	0.00	0.14	1.00	0.00
CNN	0.61	0.01	1.00	0.13	0.05	0.94	0.00	0.00	1.00	0.15	0.07	0.93	0.48	0.02	1.00	0.20	0.03	0.98
IHT	0.20	0.03	0.98	0.15	0.59	0.39	0.14	1.00	0.00	0.14	0.96	0.03	0.14	1.00	0.00	0.14	1.00	0.00
ROS	0.12	0.01	0.99	0.16	0.07	0.95	0.14	1.00	0.00	0.00	0.00	0.99	0.11	0.10	0.88	0.13	0.59	0.33
SMOTE B1	0.40	0.02	0.99	0.17	0.08	0.93	0.14	1.00	0.00	0.00	0.00	0.99	0.38	0.01	1.00	0.14	0.31	0.69
SMOTE B2	0.39	0.02	0.99	0.15	0.07	0.93	0.14	1.00	0.00	0.00	0.00	0.99	0.20	0.03	0.98	0.14	0.41	0.58
SMOTE SVM	0.29	0.02	0.99	0.12	0.06	0.93	0.11	0.80	0.20	0.06	0.01	0.98	0.28	0.03	0.99	0.13	0.83	0.11
ADASYN	0.35	0.02	0.99	0.17	0.10	0.92	0.00	0.00	1.00	0.00	0.00	0.99	0.35	0.10	0.97	0.13	0.46	0.51
SMOTE	0.16	0.01	0.99	0.14	0.07	0.93	0.00	0.00	1.00	0.00	0.00	1.00	0.12	0.09	0.89	0.70	0.01	1.00
SMOTE Tomek	0.19	0.02	0.99	0.17	0.07	0.94	0.14	0.99	0.01	0.00	0.00	0.99	0.43	0.01	1.00	0.16	0.26	0.77
SMOTE ENN	0.37	0.02	0.99	0.15	0.07	0.93	0.00	0.00	1.00	0.00	0.00	0.99	0.40	0.01	1.00	0.17	0.14	0.89

TABLE VI. EXPERIMENT 2 F1-SCORE (F) AND G MEAN (G) OF DIFFERENT CLASSIFICATION METHODS USING DIFFERENT RESAMPLING

	RFC		DTC		SVC		LSVC		BNB		NC	
	F1	***G***	***F1***	***G***	***F1***	***G***	***F1***	***G***	***F1***	***G***	***F1***	***G***
No Sampling	0.01	0.03	0.10	0.26	0.00	0.00	0.00	0.00	0.22	0.38	0.23	0.37
Tomek Links	0.01	0.06	0.09	0.26	0.00	0.00	0.00	0.00	0.22	0.38	0.24	0.37
OSS	0.01	0.05	0.09	0.26	0.00	0.00	0.00	0.00	0.22	0.38	0.23	0.40
NCR	0.01	0.02	0.11	0.28	0.00	0.00	0.00	0.00	0.22	0.38	0.24	0.35
ENN	0.01	0.06	0.10	0.27	0.00	0.00	0.00	0.00	0.22	0.38	0.24	0.36
RENN	0.01	0.03	0.10	0.27	0.00	0.00	0.00	0.00	0.22	0.38	0.24	0.36
RUS	0.03	0.11	0.13	0.26	0.25	0.01	0.23	0.46	0.19	0.40	0.24	0.13
CC	0.01	0.07	0.08	0.23	0.24	0.11	0.22	0.50	0.25	0.03	0.22	0.32
NM	0.01	0.03	0.08	0.22	0.25	0.00	0.25	0.00	0.25	0.00	0.25	0.00
CNN	0.02	0.10	0.07	0.22	0.00	0.00	0.10	0.26	0.04	0.14	0.05	0.17
IHT	0.05	0.15	0.18	0.18	0.25	0.00	0.24	0.14	0.25	0.03	0.25	0.01
ROS	0.02	0.09	0.09	0.25	0.25	0.00	0.00	0.00	0.10	0.29	0.21	0.44
SMOTE B1	0.04	0.14	0.11	0.27	0.25	0.03	0.00	0.00	0.01	0.08	0.19	0.46
SMOTE B2	0.05	0.15	0.09	0.25	0.25	0.01	0.00	0.00	0.06	0.18	0.21	0.48
SMOTE SVM	0.04	0.14	0.08	0.23	0.20	0.00	0.01	0.07	0.05	0.17	0.23	0.30
ADASYN	0.04	0.15	0.13	0.30	0.00	0.00	0.00	0.00	0.15	0.30	0.21	0.48
SMOTE	0.01	0.07	0.10	0.26	0.00	0.00	0.00	0.00	0.10	0.29	0.01	0.07
SMOTE Tomek	0.03	0.11	0.10	0.26	0.24	0.06	0.00	0.00	0.02	0.10	0.19	0.45
SMOTE ENN	0.04	0.14	0.10	0.26	0.00	0.00	0.00	0.00	0.02	0.10	0.15	0.34

REFERENCES

[1] A. Shakiba, R. Green, and R. Dyer, "FourD: do developers discuss design? revisited," in *Proceedings of the 2nd International Workshop on Software Analytics - SWAN 2016*, Seattle, Washington, 2016, pp. 43–46.

[2] P. Hart, "The condensed nearest neighbor rule (Corresp.)," *IEEE Trans. Inf. Theory*, vol. 14, no. 3, pp. 515–516, 1968.

[3] I. Tomek, "Two Modifications of CNN," *IEEE Trans. Syst. Man Cybern.*, vol. 6, no. 11, pp. 769–772, 1976.

[4] G. Lemaitre, F. Nogueira, and C. K. Aridas, "Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning," *arXiv [cs.LG]*, 21-Sep-2016.

[5] I. Mani and I. Zhang, "kNN approach to unbalanced data distributions: a case study involving information extraction," in *Proceedings of workshop on learning from imbalanced datasets*, 2003.

[6] M. Kubat, S. Matwin, and Others, "Addressing the curse of imbalanced training sets: one-sided selection," in *ICML*, 1997, vol. 97, pp. 179–186.

[7] J. Laurikkala, "Improving Identification of Difficult Small Classes by Balancing Class Distribution," in *Artificial Intelligence in Medicine*, 2001, pp. 63–66.

[8] D. L. Wilson, "Asymptotic Properties of Nearest Neighbor Rules Using Edited Data," *IEEE Trans. Syst. Man Cybern.*, vol. 2, no. 3, pp. 408–421, 1972.

[9] I. Tomek, "An experiment with the edited nearest-neighbor rule," *IEEE Trans. Syst. Man Cybern.*, vol. SMC-6, no. 6, pp. 448–452, 1976.

[10] M. R. Smith, T. Martinez, and C. Giraud-Carrier, "An instance level analysis of data complexity," *Mach. Learn.*, vol. 95, no. 2, pp. 225–256, Nov. 2013.

[11] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, vol. 16, pp. 321–357, 2002.

[12] H. Han, W.-Y. Wang, and B.-H. Mao, "Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning," in *Advances in Intelligent Computing*, 2005, pp. 878–887.

[13] H. M. Nguyen, E. W. Cooper, and K. Kamei, "Borderline over-sampling for imbalanced data classification," *Int. J. Knowl. Eng. Soft Data Paradig.*, vol. 3, no. 1, pp. 4–21, 2011.

[14] Haibo He, H. He, Y. Bai, E. A. Garcia, and S. Li, "ADASYN: Adaptive synthetic sampling approach for imbalanced learning," in *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, 2008.

[15] G. E. Batista, A. L. C. Bazzan, and M. C. Monard, "Balancing Training Data for Automated Annotation of Keywords: a Case Study," in *WOB*, 2003, pp. 10–18.

[16] G. E. A. P. A. Batista, R. C. Prati, and M. C. Monard, "A study of the behavior of several methods for balancing machine learning training data," *ACM SIGKDD Explorations Newsletter*, vol. 6, no. 1, pp. 20–29, Jun. 2004.

[17] P. Baldi, S. Brunak, Y. Chauvin, C. A. Andersen, and H. Nielsen, "Assessing the accuracy of prediction algorithms for classification: an overview," *Bioinformatics*, vol. 16, no. 5, pp. 412–424, May 2000.

[18] M.-J. Kim, D.-K. Kang, and H. B. Kim, "Geometric mean based boosting algorithm with over-sampling to resolve data imbalance problem for bankruptcy prediction," *Expert Syst. Appl.*, vol. 42, no. 3, pp. 1074–1082, 2015.

[19] F. Pedregosa *et al.*, "Scikit-learn: Machine Learning in Python," *J. Mach. Learn. Res.*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[20] L. Breiman, "Random Forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, Oct. 2001.

[21] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification and regression trees*. CRC press, 1984.
[22] A. J. Smola and B. Schölkopf, “A tutorial on support vector regression,” *Stat. Comput.*, vol. 14, no. 3, pp. 199–222, Aug. 2004.
[23] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, “LIBLINEAR: A Library for Large Linear Classification,” *J. Mach. Learn. Res.*, vol. 9, no. Aug, pp. 1871–1874, 2008.
[24] A. McCallum, K. Nigam, and Others, “A comparison of event models for naive bayes text classification,” in *AAAI-98 workshop on learning for text categorization*, 1998, vol. 752, pp. 41–48.
[25] R. Tibshirani, T. Hastie, B. Narasimhan, and G. Chu, “Diagnosis of multiple cancer types by shrunken centroids of gene expression,” *Proc. Natl. Acad. Sci. U. S. A.*, vol. 99, no. 10, pp. 6567–6572, May 2002.
[26] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*, vol. 100. 2008, pp. 234–265.
[27] X.-Y. Liu, J. Wu, and Z.-H. Zhou, “Exploratory undersampling for class-imbalance learning,” *IEEE Trans. Syst. Man Cybern. B Cybern.*, vol. 39, no. 2, pp. 539–550, Apr. 2009.