Impact of Label Noise and Efficacy of Noise Filters in Software Defect Prediction

Shihab Shahriar Khan, Nishat Tasnim Niloy, Md. Aquib Azmain, Ahmedul Kabir
Institute of Information Technology
University of Dhaka
Dhaka, Bangladesh
(bsse0703,bsse0723,bsse0718,kabir)@iit.du.ac.bd

***Abstract*—A well-established fact in the domain of software defect classification is that dataset labels collected using automated algorithms contain noise. Several prior studies examined the impact of noise and proposed novel ways of dealing with this issue. Those studies, however, relied on randomly simulated artificial noise on clean datasets, but real-world noise is not random. Using a recently proposed dataset with both clean labels annotated by experts and noisy labels obtained by heuristics, this paper revisits the question of how label noise impacts the defect classification performance and demonstrate how the answer varies among several types of classification algorithms. Based on a diverse set of 9 different noise filters, this paper empirically investigates their ability to improve the performance of classifiers trained with label noise. Contrary to previous findings, we observe that the noise filters mostly struggle to improve performance over unfiltered noisy data. Lastly, we conduct several small-scale experiments in a bid to explain our findings and uncover actionable insights.**

***Index Terms*—Defect prediction, Label Noise, Class Imbalance, Noise detection**

I. Introduction

Like any machine learning task, the quality of defect prediction models depends highly on the quality of data used to train them. Unfortunately, noise is pervasive in binary defect prediction datasets, particularly label noise [1], [2]. Label noise occurs when a defective artifact is labeled as non-defective, or vice-versa. There are many possible sources of such noise: mislabeling of issues [3], failure to link issues to relevant code changes [4] or bias of human annotators [5].

Several studies suggest that the presence of mislabeled samples can significantly impact the performance of defect prediction models [1], [6]. Studying the impact of noise is a tricky problem, however, as it requires access to a set of samples that are known to be reliable. Evaluating the model on noisy test data might provide an unreliable estimate of its generalizability. A common approach is to first clean data [7], [8] or to only choose datasets believed to be of high-quality [5], [6], and then randomly introduce artificial noise only in the training subset. But the label noise of defect prediction dataset is not random, and random noise tends to overestimate noise's impact compared to naturally occurring noise [9].

According to [10], the gold standard for label noise research is to use a dataset where both naturally occurring noisy labels, and their clean, reliable counterparts (produced for example with the help of domain experts) are available. Recently, Yatish *el al.* [11] presented such a defect prediction dataset, and concluded, somewhat surprisingly, that the impact of realistic noise on classifier performance is "modest". To resolve this apparent disparity between results from artificial and clean noise, using the same dataset as [11], this paper revisits the question of how noise impacts classifier performance. We show that except for Naive Bayes, noise has a high impact on most classifiers. Decision tree and random forest were found to lose comparatively the most performance.

Another objective is to find out whether this loss of performance can be mitigated by noise handling techniques. There are several ways to handle dataset noise such as cost-sensitive learning [12], robust classifiers [13] or noise filters. We restrict ourselves to noise filter in this study, and most of the noise handling approaches proposed in defect prediction literature fall under this category. In this paper, 9 filters of diverse properties are investigated to analyze how they individually perform, and how classifiers react to them. To the best of our knowledge, this is the first paper to empirically compare such a broad array of filters in defect prediction setting.

The result of the application of filter is somewhat mixed. On the one hand, across all classifiers and datasets, noise filters barely improve performance over unfiltered noisy labels. On the other hand, from a classifier standpoint, performance improves quite significantly when *best* filter performance is considered. The takeaway is that while filtering *can* improve performance, it crucially depends on the right combination of classifier and filter.

Our final objective is to explain our findings and extract actionable insights from them. We conduct several small-scale experiments to that end. These revealed that the noise model we study is far more challenging than artificially introduced random noise, which may explain the observed big impact of noise on classifiers. They also reveal how traditional approaches to fight the natural imbalance of defect datasets is particularly inadequate in the high-level noisy setting that we study.

II. Study Design

The scope of this paper is limited to the "within-project" post-release defect prediction scenario, and the main focus

DOI reference number: 10.18293/SEKE2020-126.

here is on prediction, not interpretation. Throughout the paper, the noise level (NL) of a dataset refers to the percentage of its total samples that are mislabeled. Imbalance ratio (IR) refers to the ratio of the number of samples in the majority (non-defective) class to the number of samples in minority (defective) class for clean labels, nIR does the same but for noisy labels. The term "model" refers to an instance of the combination of classifier, imbalance-method and filters used. P→N denotes fraction of originally positive (defective) samples that have been flipped to negative (non-defective) in heuristic-based labeling, and N→P denotes the opposite.

All mentions of averages in this study are actually trimmed mean, calculated after trimming away 5% of extreme values from each side. Wilcoxon signed-rank test [14] is used to test statistical significance, with the p-value set to .01. Benjamini-Hochberg procedure is used to correct for multiple comparisons. To compute the effect size, Hedges' g [15] is used with an interpretation of values according to [16], that is $|g| < 0.2$ "negligible", $|g| < 0.5$ "small", $|g| < 0.8$ "medium", otherwise "large"

A. Dataset Description:

This paper uses 32 datasets from 9 open-source software systems presented in [11]. To identify defects introduced by a release, traditionally used heuristic methods rely on strong assumptions like "all bugs reported after release X is introduced in release X" or that "all defect-fixing commits that affect the release X occur within (say) 6 months after it's release", etc. Furthermore, to link a bug report to a defect-fixing commit, they assume that the logs of all bug-fixing commits contain a specific set of keywords (e.g. Bugs, Fix etc.) along with relevant issue ID.

In contrast, datasets used here are all collected from JIRA issue tracker, which allows developers of a software to define what releases were affected by a given bug, earliest of which can be assumed to be the one that introduced it. This makes the produced defective/non-defective labels comparatively far more reliable. For the 32 datasets studied here, summary statistics for their key characteristics is presented in Table 1. #ND and #defective denote number of non-defective and defective samples respectively in clean labels.

TABLE I: Summary statistics of key dataset characteristics

Dataset Property	Min	Median	Max
Size	731	1717	8846
NL(%)	2.35	13.32	28.99
IR	1.97	8.24	45.07
nIR	3.09	12.65	56.87
#defective	26	197	669
#ND	609	1455	8654
P→N(%)	19.23	63.27	93.43
N→P(%)	1.05	4.12	20.97

B. Classifiers

For subsequent experiments, 6 classifiers have been used: Decision Tree (DT), K Nearest Neighbor (KNN), Naive Bayes (NB), Logistic Regression (LR), Support Vector Machine (SVM) and Random Forest (RF). Each of these classifiers has been combined with 5 data balancing techniques: Wilson's editing (ENN), Random Under-Sampling (RUS), SMOTE, Tomek Links (TOMEK) and None (representing no sampling), totaling $6*5 = 30$ models. Two bagging classifiers using Naive Bayes and Decision Tree are also used as base learners, both of which apply RUS independently at each base learner (BagNB and BagDT). This brings the total number of models studied to 32.

C. Noise Filters

All the 9 noise filters operate under the assumption that mislabeled instances are harder to predict than clean ones. These methods can be roughly divided into two groups. The first group, ensemble-based methods, assume a mislabeled training instance will be frequently misclassified by a committee of classifiers. Iterative Partitioning Filter (IPF) [17], Random Forest Filter (IHF) [18] and Instance Hardness Threshold (IHT) [19] belong to this group. IPF partitions the training data into n subsets to train a decision tree on each, and all n classifiers are used to predict the label of each instance (and thereby detect mislabeling) in the training set. RFF takes a cross-validation approach- it uses $n-1$ subsets of training data to train a random forest classifier, and uses its base trees to predict on remaining subset. IHT differs from IHF only in that it down-samples the majority class, leaving minority class untouched.

The second group uses an instance's nearest neighbors to predict its label. This group includes Neighborhood Cleaning Rule (NCL) [20], SPIDER2 [21], Edited Nearest Neighbor (ENN) [22], SMOTE_ENN [23] and SMOTE_IPF [24] and Closest List Noise Identification (CLNI) [6]. CLNI finds for each instance the percentage of its K nearest neighbors that have different class values, instances for which this percentage cross a certain threshold are removed and repeats this procedure unless some stopping criterion is met. For space consideration, the discussion about the rest of the filters is omitted.

Six of these filters belong to the family of focused imbalance-methods- noise identification and removal are integrated into them. Since the noise detectors have a tendency to overestimate the noise likelihood of minority class [25], for the rest 3 types- IPF, CLNI and RFF, this paper relies on the previously discussed imbalance-methods to balance dataset before applying filtering. All these filters are compared against the baseline "NoF", meaning no filtering.

D. Evaluation:

Following the guidelines of [26], our primary choice for a model's performance evaluation is Matthews Correlation Coefficient (MCC). For a given confusion matrix, MCC is defined as:

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$$

Since this is threshold-dependent, the area under the Precision-Recall curve (APRC) is also used to complement MCC, but we note its performance only when APRC result diverges significantly from MCC. APRC is used over the more popular alternative area under the receiver operating characteristic curve (roc-auc) following the recommendation of [27].

In all of the following experiments, only clean labels are used for evaluation. 5-fold cross-validation has been repeated 5 times for measuring all reported performance values. The experiments are conducted with the help of scikit-learn [28] and imbalanced-learn [29] libraries. For all classifiers, the default hyper-parameter values provided in these libraries are used. The dataset and source code of this study can be found online [1].

III. Results & Discussion

A. How does the presence of label noise impact bug detection?

Figure 1 shows how the performance of examined classifiers compare between clean and noisy labels. As expected, all classifiers fare worse when noise is included. But some, for example, decision tree (DT), reacts quite poorly to noise. This is expected since unpruned decision trees are known to easily overfit [30]. But performance loss of random forest (RF) was somewhat surprising as it has been frequently found to be noise resistant in previous studies [31], [32]. On the opposite spectrum is Naive Bayes and its balanced bagging version BagNB. In fact, as Table II shows, these are the only two classifiers where label noise seems to have had a small effect in terms of MCC. Among others, only SVM has a medium effect, all else are highly impacted by noise. The impact is comparatively less severe for APRC, which is expected since it's threshold-independent. But overall, the impact is statistically significant for all of the results in Table II except one. This result contradicts [2] or is slightly at odds with [11] previous studies, which reported noisy labels having an only modest effect on classifiers.

This performance of Naive Bayes under noise is consistent with previous studies [33] that demonstrated Naive Bayes' superiority in the defect classification task. This is slightly at odds with the findings from general-purpose (i.e. not domain-specific) datasets like the ones from UCI [34], where random Forest, SVM or Boosting approaches perform well. As noise label is pretty ubiquitous in bug prediction datasets used in those studies [7], [8], this finding suggests that Naive Bayes' particular success in defect classification domain may have been primarily due to its unique robustness in the face of label noise.

To summarize:

- The impact of label noise is statistically significant for all classifiers, and the impact is quite large for most.
- Random forest and similar balanced bagging with decision tree perform best when trained with clean labels.

[1]https://figshare.com/s/372afb62060475b91e9e

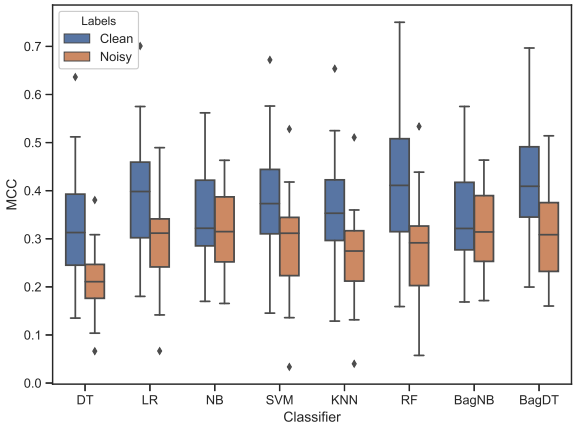


Fig. 1: Comparison of performances of classifiers trained with clean and noisy labels

- NB and bagging with NB are two of the best performing classifiers with noisy labels, particularly owing to NB's surprisingly good inherent robustness to noise.

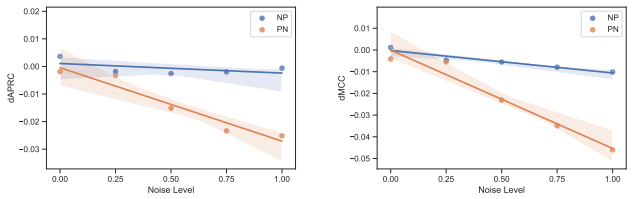
Discussion: Why is the impact of noise high? One obvious explanation is that the amount of noise in studied datasets is comparatively high compared to previous studies. All of the filters that we studied were tested with the assumption that noise level will not cross 50%. Even many theoretical guarantees we have about learning from corrupted labels is grounded on the same assumption [12], [35]. But as Table I shows, at least for defective class, this assumption does not hold. In 23/32 datasets, noise level in defective class (P→N) is higher than that 50% mark. While overall noise level is well below 50% for all the datasets, several prior studies show using artificial noise that the P→N noise is relatively far more harmful than N→P.

We demonstrate this in a slightly different way: we start with clean labels, for each class we select the samples that are mislabeled by heuristic method, and then flip a certain percentage of them, while keeping the labels of other class

TABLE II: Effect of label noise on classifier performance. "Delta" denotes average *loss* of performance across all datasets and imbalance methods. Results that are NOT statistically significant are marked with *

	MCC			APRC		
Classifier	Delta	Effect	Interpret.	Delta	Effect	Interpret.
DT	.116	1.312	Large	0.082	0.611	Medium
LR	.103	1.003	Large	0.079	0.515	Medium
NB	.022	0.277	Small	0.013	0.125	Neglig.
SVM	.081	0.755	Medium	0.089	0.551	Medium
KNN	.097	1.000	Large	0.086	0.607	Medium
RF	.141	1.224	Large	0.118	0.760	Medium
BagNB	.021	0.268	Small	0.011*	0.087	Neglig.
BagDT	.097	0.962	Large	0.097	0.586	Medium

constant. We then compute how this two types of intentionally introduced noise degrades performance w.r.t clean data, aggregated over all datasets and models.



(a) Impact of each noise type on APRC (b) Impact of each noise type on MCC

Fig. 2: Impact of P→N vs N→P noise

As Figure 2 shows, performance decreases much more rapidly with P→N noise than N→P. This implies where noise occurs is more important than overall noise level, and that's why these datasets pose a very difficult task for any learner.

B. How much can noise filters recover performance?

As previously mentioned, 9 noise filters are applied to the training dataset. Figure 3 boxplot shows how much each of them improves performance over unfiltered noisy labels, i.e the difference in MCC and APRC, across 32 models and 32 datasets.

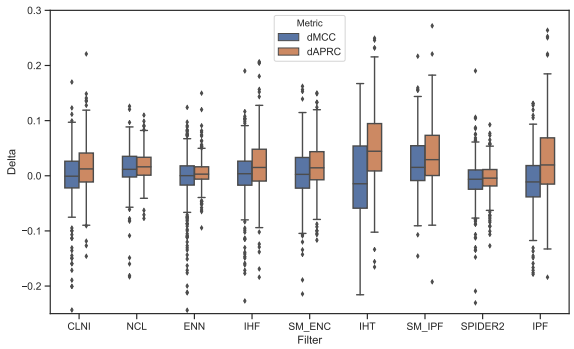


Fig. 3: Performance *improvement* by noise filters on noisy labels across all models and datasets w.r.t baseline no-filering (NoF)

One salient finding is that the median for most of the filters lies pretty close to zero, a value that indicates no improvement over non-filtered data. In fact, the boxes quite often go below the zero mark, these are the cases where filtering actually decreased performance. Some outliers go even below the range presented in the figure.

As both Figure 3 and Table III show, the results vary slightly among the evaluation metrics. Among all the filters, classifiers and datasets, APRC performance on average is improved by .0203, for MCC this value is 0.0012. With MCC, only 2 out of 9 filters, NCL and SMOTE_IPF, bring statistically significant improvement. With APRC, this is true for all except SPIDER2. But irrespective of evaluation metric, the effect of performance improvement remains small even for the best performing filters.

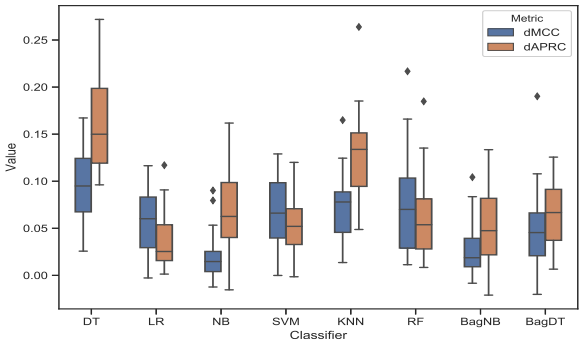


Fig. 4: Performance improvement of each classifier using the best filter for that classifier

In Figure 4, for each classifier, we plot the improvement brought by the best among all 9 filters for that classifier. This represents an upper bound on how much a classifier can recover performance from the application of filtering. The figure reveals that KNN and DT classifiers can benefit most from noise filters. Using only the best value among filters, performance improvement across all models and datasets on average is .057 for MCC and .077 for APRC. This implies filtering *can* positively impact performance, as long as right classifier-filter combination is used.

***Discussion**:* So, Why do filters struggle to improve performance? Apart from high noise in defective class discussed before, we believe a big part of this answer lies in the fact that these datasets are highly class-imbalanced. Next, we consider 3 of the most common ways to address imbalance in turn: no sampling, under-sampling and over-sampling, and provide preliminary evidence to demonstrate why they might be inadequate for the datasets at hand.

1) No Sampling: We begin with the baseline where we do not do anything to address class imbalance. [26] suggests that class balancing is important for moderate or highly imbalanced datasets, where moderate imbalance was defined as having

TABLE III: Performance *improvement* for each noise filter across all datasets and models. Statistically significant improvements are marked with *.

Filter	MCC Delta	MCC Effect	MCC Interpr.	APRC Delta	APRC Effect	APRC Interpr.
SM_IPF	0.0237*	0.275	Small	0.0359*	0.274	Small
SPIDER2	-0.0080	-0.121	Neglig.	-0.0046	-0.04	Neglig.
IHF	0.0001	-0.061	Neglig.	0.0188*	0.138	Neglig.
NCL	0.0163*	0.156	Neglig.	0.0172*	0.127	Neglig.
ENN	-0.0042	-0.079	Neglig.	0.0057*	0.045	Neglig.
IHT	-0.0066	-0.079	Neglig.	0.0496*	0.356	Small
CLNI	-0.0031	-0.099	Neglig.	0.0159*	0.119	Neglig.
IPF	-0.0116	-0.130	Neglig.	0.0264*	0.205	Small
SM_ENC	0.0041	0.038	Neglig.	0.0175*	0.119	Neglig.

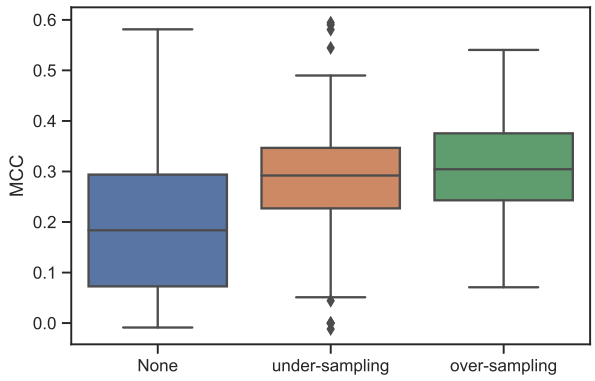


Fig. 5: Comparison of 3 approaches to data balancing accross all datasets and models. under-sampling includes ENN and RUS, over-sampling only includes SMOTE.

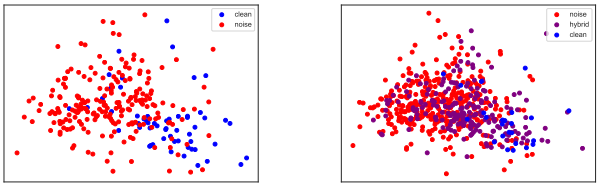
IR$>$ 3.94. Using the same threshold, 27 out of 32 datasets we use is moderately or highly imbalanced, suggesting this option might not be optimal. We illustrate this using IPF, one of the filters which does not have data balancing baked in. As Figure 5 shows, both of the alternatives clearly outperform no-sampling.

2) Under-Sampling: These techniques remove samples only from majority class, unless a desired level of balance with minority class is reached. One problem with these approaches is that when imbalance is high, this means throwing out a big portion of dataset. For example, given our median nIR=12.65, under-samplers will have to throw out 85.4% of overall data to create perfect balance.

Some techniques try to minimize the impact of data loss by filtering out uninformative or noisy samples. In fact, 3 of our 9 filters: IHT, NCL and ENN, belong to this group. As we'll show, even when they work exactly as intended i.e. even in the best case scenario, they still struggle to noticeably improve performance over simple random under-sampling. To test this, we first removed each mislabeled sample from majority class of training set, and then kept randomly removing samples until class balance is achieved. While the improvement (across all datasets and models) over random under-sampling is statistically significant, effect size reveals that the improvement is "negligible" for both MCC and APRC (.155 and .129 respectively).

3) Over-Sampling: As a representative of this family of samplers, we choose perhaps the most widely used data balancing technique: SMOTE [36]. In short, SMOTE randomly selects one of the minority (defective) samples and one of it's nearest neighbors, then samples a random point from the line connecting those two samples in feature space. This new sample naturally gets labeled defective. Therefore, quality of labels of new samples depends crucially on the quality of existing minority samples' labels.

Unfortunately, majority of the samples labeled as defective are actually non-defective due to noise. While the ratio of N$\rightarrow$P noise as shown in Table I is comparatively small, even that small noise level can overwhelm minority class due to high imbalance. For example, in a dataset with nIR=12.65, P$\rightarrow$N=63.27% and N$\rightarrow$P=4.12% (all are median values taken from Table I), only 41.3% of samples labeled as defective will actually be defective. This in turn means that only 17% of SMOTE-generated defective samples is expected to originate from a pair of defective samples, whereas for around 34% samples both of their parents will actually be non-defective. About half ($\sim$ 49%) will originate from one clean and one mislabeled sample. Using PCA transformation, we illustrate this idea with JRuby-1.5.0 dataset in Figure 6.



(a) Original defective samples (b) Samples generated by SMOTE

Fig. 6: Distribution of mislabeled and clean samples in defective class. Explained variance for the figures is 57% and 59% respectively.

IV. Threats to Validity

Our study crucially depends on the assumption that the labels we assume to be clean are actually clean. One issue is that these labels are human-annotated. While this makes the so-called clean labels far more reliable than their heuristics-based counterparts, they are still susceptible to human error, due to phenomenons like snoring [37] or annotator bias [5] for example.

Also, a few prominent classifiers like Neural Network or XGBoost [38] are excluded in this study, mainly due to their computationally extensive training procedures. Similar consideration also led us to skip extensive hyper-parameter tuning, something that can impact a classifier's performance [39]. We note however, that any such optimization would have to be carried out on a subset of noisy training data, and this can lead to poor choice of hyper-parameter values [40].

V. Conclusion and Future Work

By using a diverse set of classifiers, imbalance-methods and noise filters, this study empirically investigates how the presence of label noise in post-release defect prediction datasets affect performance and evaluates the effectiveness of noise filters in minimizing the adverse effects of noise. The principal conclusions of this study are (1) Label noise in bug prediction datasets do have large impact on most classifier's performance, (2) Noise filtering isn't guaranteed to improve performance, but with the right choice of classifier-filter combination, it can yield significant improvement especially for classifiers

that easily overfit, e.g., decision tree. This study also revealed the highly robust nature of the Naive Bayes algorithm, the surprising brittleness of Random Forest and took the first steps towards explaining these findings.

For future work, we plan to investigate several alternatives to filtering for noise handling. The relatively higher cost of P→N noise suggests while designing any auto defect-labeling algorithm, recall of defect class should be prioritized over precision. We also plan to explore this idea in future.

REFERENCES

[1] C. Bird, A. Bachmann, E. Aune, J. Duffy, A. Bernstein, V. Filkov, and P. Devanbu, "Fair and balanced?" in *Proceedings of the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering on European software engineering conference and foundations of software engineering symposium - ESEC/FSE '09*. ACM Press, 2009. [Online]. Available: https://doi.org/10.1145/1595696.1595716

[2] K. Herzig, S. Just, and A. Zeller, "It's not a bug, it's a feature: how misclassification impacts bug prediction," in *Proceedings of the 2013 international conference on software engineering*. IEEE Press, 2013, pp. 392–401.

[3] J. Aranda and G. Venolia, "The secret life of bugs: Going past the errors and omissions in software repositories," in *Proceedings of the 31st international conference on software engineering*. IEEE Computer Society, 2009, pp. 298–308.

[4] A. Bachmann, C. Bird, F. Rahman, P. Devanbu, and A. Bernstein, "The missing links: bugs and bug-fix commits," in *Proceedings of the eighteenth ACM SIGSOFT international symposium on Foundations of software engineering*. ACM, 2010, pp. 97–106.

[5] F. Rahman, D. Posnett, I. Herraiz, and P. Devanbu, "Sample size vs. bias in defect prediction," in *Proceedings of the 2013 9th joint meeting on foundations of software engineering*. ACM, 2013, pp. 147–157.

[6] S. Kim, H. Zhang, R. Wu, and L. Gong, "Dealing with noise in defect prediction," in *2011 33rd International Conference on Software Engineering (ICSE)*. IEEE, 2011, pp. 481–490.

[7] J. Van Hulse, T. M. Khoshgoftaar, and A. Napolitano, "Skewed class distributions and mislabeled examples," in *Seventh IEEE International Conference on Data Mining Workshops (ICDMW 2007)*. IEEE, 2007, pp. 477–482.

[8] C. Seiffert, T. M. Khoshgoftaar, J. Van Hulse, and A. Folleco, "An empirical study of the classification performance of learners on imbalanced and noisy software quality data," *Information Sciences*, vol. 259, pp. 571–595, 2014.

[9] C. Tantithamthavorn, S. McIntosh, A. E. Hassan, A. Ihara, and K. Matsumoto, "The impact of mislabelling on the performance and interpretation of defect prediction models," in *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, vol. 1. IEEE, 2015, pp. 812–823.

[10] B. Frénay and M. Verleysen, "Classification in the presence of label noise: a survey," *IEEE transactions on neural networks and learning systems*, vol. 25, no. 5, pp. 845–869, 2013.

[11] S. Yatish, J. Jiarpakdee, P. Thongtanunam, and C. Tantithamthavorn, "Mining software defects: should we consider affected releases?" in *Proceedings of the 41st International Conference on Software Engineering*. IEEE Press, 2019, pp. 654–665.

[12] N. Natarajan, I. S. Dhillon, P. Ravikumar, and A. Tewari, "Cost-sensitive learning with noisy labels." *Journal of Machine Learning Research*, vol. 18, pp. 155–1, 2017.

[13] T. G. Dietterich, "An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization," *Machine learning*, vol. 40, no. 2, pp. 139–157, 2000.

[14] R. Woolson, "Wilcoxon signed-rank test," *Wiley encyclopedia of clinical trials*, pp. 1–3, 2007.

[15] L. V. Hedges and I. Olkin, *Statistical methods for meta-analysis*. Academic press, 2014.

[16] J. Cohen, "A power primer." *Psychological bulletin*, vol. 112, no. 1, p. 155, 1992.

[17] T. M. Khoshgoftaar and P. Rebours, "Improving software quality prediction by noise filtering techniques," *Journal of Computer Science and Technology*, vol. 22, no. 3, pp. 387–396, 2007.

[18] M. Sabzevari, G. Martínez-Muñoz, and A. Suárez, "A two-stage ensemble method for the detection of class-label noise," *Neurocomputing*, vol. 275, pp. 2374–2383, 2018.

[19] M. R. Smith, T. Martinez, and C. Giraud-Carrier, "An instance level analysis of data complexity," *Machine learning*, vol. 95, no. 2, pp. 225–256, 2014.

[20] J. Laurikkala, "Improving identification of difficult small classes by balancing class distribution," in *Conference on Artificial Intelligence in Medicine in Europe*. Springer, 2001, pp. 63–66.

[21] K. Napierała, J. Stefanowski, and S. Wilk, "Learning from imbalanced data in presence of noisy and borderline examples," in *International Conference on Rough Sets and Current Trends in Computing*. Springer, 2010, pp. 158–167.

[22] D. L. Wilson, "Asymptotic properties of nearest neighbor rules using edited data," *IEEE Transactions on Systems, Man, and Cybernetics*, no. 3, pp. 408–421, 1972.

[23] G. E. 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, 2004.

[24] J. A. Sáez, J. Luengo, J. Stefanowski, and F. Herrera, "Smote–ipf: Addressing the noisy and borderline examples problem in imbalanced classification by a re-sampling method with filtering," *Information Sciences*, vol. 291, pp. 184–203, 2015.

[25] J. Van Hulse and T. Khoshgoftaar, "Knowledge discovery from imbalanced and noisy data," *Data & Knowledge Engineering*, vol. 68, no. 12, pp. 1513–1542, 2009.

[26] Q. Song, Y. Guo, and M. Shepperd, "A comprehensive investigation of the role of imbalanced learning for software defect prediction," *IEEE Transactions on Software Engineering*, 2018.

[27] T. Saito and M. Rehmsmeier, "The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets," *PloS one*, vol. 10, no. 3, p. e0118432, 2015.

[28] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *Journal of machine learning research*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[29] G. Lemaître, F. Nogueira, and C. K. Aridas, "Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 559–563, 2017.

[30] M. Bramer, "Using j-pruning to reduce overfitting in classification trees," in *Research and Development in Intelligent Systems XVIII*. Springer, 2002, pp. 25–38.

[31] T. M. Khoshgoftaar, J. Van Hulse, and A. Napolitano, "Comparing boosting and bagging techniques with noisy and imbalanced data," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 41, no. 3, pp. 552–568, 2010.

[32] C. Pelletier, S. Valero, J. Inglada, N. Champion, C. Marais Sicre, and G. Dedieu, "Effect of training class label noise on classification performances for land cover mapping with satellite image time series," *Remote Sensing*, vol. 9, no. 2, p. 173, 2017.

[33] S. Wang and X. Yao, "Using class imbalance learning for software defect prediction," *IEEE Transactions on Reliability*, vol. 62, no. 2, pp. 434–443, 2013.

[34] A. Asuncion and D. Newman, "Uci machine learning repository," 2007.

[35] N. Manwani and P. Sastry, "Noise tolerance under risk minimization," *IEEE transactions on cybernetics*, vol. 43, no. 3, pp. 1146–1151, 2013.

[36] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[37] A. Ahluwalia, D. Falessi, and M. Di Penta, "Snoring: a noise in defect prediction datasets," in *Proceedings of the 16th International Conference on Mining Software Repositories*. IEEE Press, 2019, pp. 63–67.

[38] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. ACM, 2016, pp. 785–794.

[39] C. Tantithamthavorn, S. McIntosh, A. E. Hassan, and K. Matsumoto, "The impact of automated parameter optimization on defect prediction models," *IEEE Transactions on Software Engineering*, vol. 45, no. 7, pp. 683–711, 2018.

[40] D. I. Inouye, P. Ravikumar, P. Das, and A. Dutta, "Hyperparameter selection under localized label noise via corrupt validation," in *NIPS Workshop*, 2017.