ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/364098519

Why Don't XAI Techniques Agree? Characterizing the Disagreements Between Post-hoc Explanations of Defect Predictions

Conference Paper · October 2022

DOI: 10.1109/ICSME55016.2022.00056

CITATION
1

READS
337

6 authors, including:

Learn Er
University of Saskatchewan
12 PUBLICATIONS **32** CITATIONS
SEE PROFILE

Gabriel Laberge
Polytechnique Montréal
20 PUBLICATIONS **53** CITATIONS
SEE PROFILE

Amin Nikanjam
K. N. Toosi University of Technology
57 PUBLICATIONS **242** CITATIONS
SEE PROFILE

Saikat Mondal
University of Saskatchewan
27 PUBLICATIONS **136** CITATIONS
SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Home Automation System View project

Formal Verification View project

All content following this page was uploaded by Learn Er on 02 October 2022.

The user has requested enhancement of the downloaded file.

Why Don't XAI Techniques Agree? Characterizing the Disagreements Between Post-hoc Explanations of Defect Predictions

Saumendu Roy
University of Saskatchewan, Canada
plz937@usask.ca

Gabriel Laberge
Polytechnique Montréal, Canada
gabriel.laberge@polymtl.ca

Banani Roy
University of Saskatchewan, Canada
banani.roy@usask.ca

Foutse Khomh
Polytechnique Montréal, Canada
foutse.khomh@polymtl.ca

Amin Nikanjam
Polytechnique Montréal, Canada
amin.nikanjam@polyml.ca

Saikat Mondal
University of Saskatchewan, Canada
saikat.mondal@usask.ca

***Abstract*—Machine Learning (ML) based defect prediction models can be used to improve the reliability and overall quality of software systems. However, such defect predictors might not be deployed in real applications due to the lack of transparency. Thus, recently, application of several post-hoc explanation methods (e.g., LIME and SHAP) have gained popularity. These explanation methods can offer insight by ranking features based on their importance in black box decisions. The explainability of ML techniques is reasonably novel in the Software Engineering community. However, it is still unclear whether such explainability methods genuinely help practitioners make better decisions regarding software maintenance. Recent user studies show that data scientists usually utilize multiple post-hoc explainers to understand a single model decision because of the lack of ground truth. Such a scenario causes disagreement between explainability methods and impedes drawing a conclusion. Therefore, our study first investigates three disagreement metrics between LIME and SHAP explanations of 10 defect-predictors, and exposes that disagreements regarding the rankings of feature importance are most frequent. Our findings lead us to propose a method of aggregating LIME and SHAP explanations that puts less emphasis on these disagreements while highlighting the aspect on which explanations agree.**

***Index Terms*—Empirical, Defect Prediction, eXplainable AI, LIME, SHAP, Software Maintenance**

I. Introduction

With the growing complexity of modern software [1], data-intensive application [2] there has come an increased difficulty in assuring high quality, and maintainability [1]. Thus, detecting defect-prone modules early in the development cycle is becoming primordial to better allocate limited resources for testing and maintenance. Different techniques have been proposed to manage the task of defect prediction. The most commonly known are Machine Learning (ML) based techniques [3]–[5]. However, ML models remain black boxes despite their apparent success, meaning it is hard for humans to understand the reasoning behind their decisions. This is a significant reason why eXplainable AI (XAI) methods such as LIME [6] and SHAP [7] have gained in popularity recently. These techniques provide explanations for any decisions made by a model (model-agnostic). Hence, they are a versatile tool to add to the traditional ML-based methodology for defect predictions. The usage of these techniques is reasonably recent in the Software Engineering community [8], [9], and many crucial questions are still unanswered. Notably, it is unclear whether or not post-hoc explanations can guide decision-making regarding software maintenance. Indeed, as recently noted, different post-hoc explainers can yield different (and even contradicting) explanations for the same model decisions [10]. Practitioners being shown conflicting justifications of why a given module is predicted as defect-prone may decide to disregard the explainability tools altogether. Therefore, it is critical to study the disagreement problem of post-hoc explanations in a Software Engineering context.

In this study, we investigate *three* disagreement metrics between LIME and SHAP (*Feature Agreement, Rank Agreement,* and *Sign Agreement*) in the context of defect prediction. We show that disagreements regarding the ranking among top-k features importance occur more often than disagreements on the sign (positive/negative) of their importance. Based on this insight, we propose a novel method to aggregate LIME and SHAP explanations by only presenting users with the top-k most important features according to both LIME and SHAP, and whose importance shares the same sign (positive/negative) across both explainers. This aggregation method ignores the ranking among top-k features as these ranking are shown to be inconsistent between explainers.

II. Background

When a user is employing a defect detection system, the burden of proof is always on the tool to illustrate the reasoning behind its decision [11]. Otherwise, practitioners may completely disregard predictions as false positives and stop using the detection tools. To resolve this issue, there has been a growing interest in applying techniques from XAI to the area of defect prediction [8], [12]. Indeed, in a survey conducted among developers and managers, it was noted that 82% of

2576-3148/22/$31.00 ©2022 IEEE
DOI 10.1109/ICSME55016.2022.00056

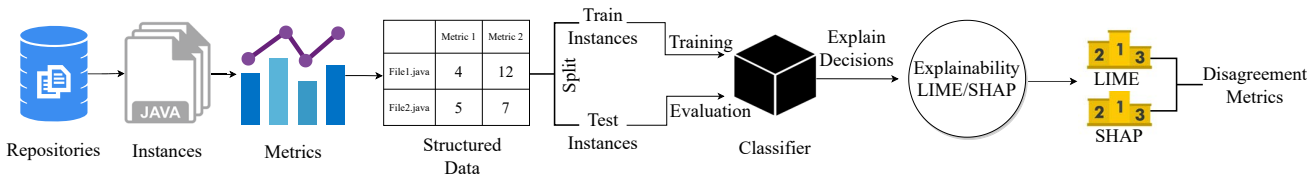


Fig. 1: Schematic diagram of the methodology.

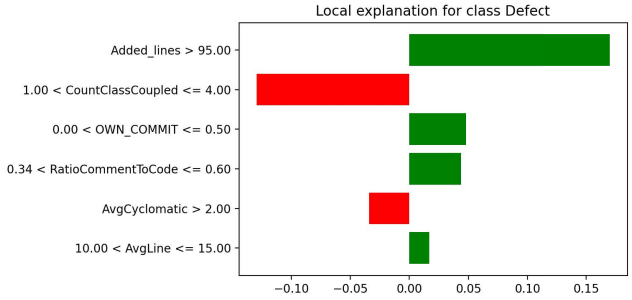


Fig. 2: Example of a LIME explanation. Each feature is attributed a score. The sign (positive or negative) represents whether or not the feature value pushes toward the defect prediction or away from it.

respondents perceived the goal of explaining defect predictions as being beneficial [13]. Jiarpakdee et al. [8] proposed to utilize model-agnostic strategies, e.g., LIME and BreakDown [14] to explain defect predictions. A modification of LIME called TimeLIME [15] was later introduced to ensure that the defect reduction plans derived from the explanations are more aligned with the past project [15]. The PyExplainer [12] was also recently proposed as an improvement of LIME that aims at making its explanations more actionable.

In this paper, we solely focus on LIME and SHAP post-hoc explanations considering the fact that they are both famous and highly cited methods and because they have open-source implementations with very simple APIs. Both LIME and SHAP explain feature importance: a positive/negative score is attributed to each feature to represent whether the value of the features pushes the prediction toward defective or away from defective. Moreover, each feature is ranked in terms of importance (i.e., the magnitude of their score), see Figure 2.

One of the fundamental problems in XAI is the lack of ground truth regarding which post-hoc explanation is the right one. Because of this limitation, it was recently noted in a user study that data scientists tend to use multiple explainable tools to understand the same model decision [10]. However, explanations from different tools need not agree; see Figure 3 for instance. In this figure, we observe three different types of disagreements between feature importance:

- The top-3 features need not be the same,
- Even if the top-3 features are the same, their importance ranking may differ,
- Even if the top-3 features are the same, the sign of their importance may not be the same.

Although here, we use top-3 features to describe disagreement, it is essential to note that we shall generally use the top-k features, where k will depend on the number of features used by the model. Krishna et al. [10] have formalized these three different notions of disagreements between feature importance via three novel metrics: namely the Feature Agreement (FA), Rank Agreement (RA), and Sign Agreement (SA), which will be discussed in details in **Section III-D**

We note that this is not the study of disagreements between post-hoc explanations of Defect Predictors. For instance, Shin et al. [16] studied how post-hoc explanations are affected by data sampling procedures and the choice of classifier. Our work is different since we focus on the difference between two explainers : LIME and SHAP, for a fixed ML model. Moreover, instead of painting a pessimistic picture by simply noting the disagreements, we aim at providing a possible aggregation mechanism to derive insight from multiple explainers even though they may not perfectly agree

III. Methodology

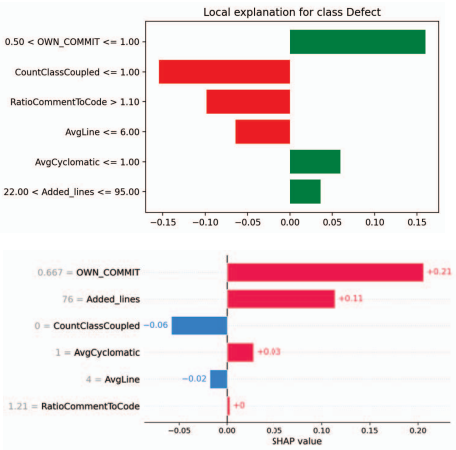
Figure 1 shows the workflow of our study. Briefly, it consists of collecting online defect prediction datasets, splitting each into training and test sets, training a ML classifier on the training set, evaluating performance on the test set, and using LIME and SHAP to explain all test set files that the model considers defective. Finally, disagreement metrics between LIME and SHAP explanations are computed.

A. Dataset Preparation

The ten defect prediction datasets we studied were accessed through a publicly-available corpus containing various versions of Java projects [17]. Said corpus contains multiple metrics about individual files from any project: 54 code metrics, five process metrics and six ownership metrics. Moreover, `Clean` or `Defect` labels are assigned to each file based on whether or not they were affected by an issue-fixing commit in the following release.

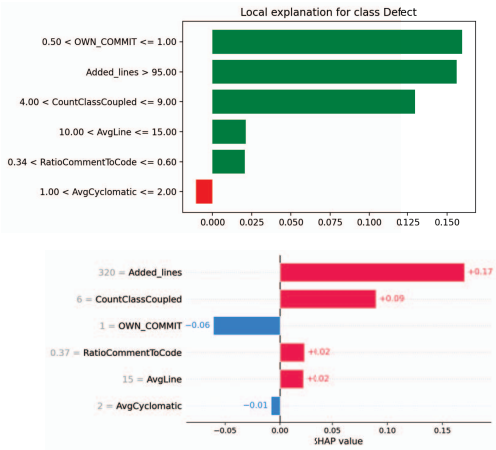
To identify the most relevant metrics for defect prediction, we performed feature selection based using the AutoSpearman method [18] as in previous studies [8], [9], [12]. A particularity of defect prediction datasets is the low ratio of defective files compared to clean ones which makes it hard to train a binary classifier accurately. To manage this issue, like previous studies [12], [15] we utilized the SMOTE algorithm, which over-samples the minority class by generating *synthetic* samples based on the k-Nearest Neighbors [19].

Example 1
Top-3 features disagree



FA=2/3, SA=2/3, RA=1/3

Example 2
Top-3 features agree
but their rank and sign disagree



FA=1,SA=2/3, RA=0

Fig. 3: Two examples of disagreements between LIME and SHAP when explaining a file predicted as defect. (Left), we have a disagreement regarding which features are the top-3 most important to explain the prediction. In fact, only two features are common in the top-3 of both LIME and SHAP. (Right), the top-3 between LIME and SHAP contain the same features, although none of these features have the same rank, and the sign of `OWN_COMMIT` is inconsistent between the two explanations.

B. Models

ML practitionners [20] and a previous studies [21] recommend the subsequent supervised ML approaches for binary classifications: Logistic Regression (LR), Random Forest (RF), Gradient Boosted Trees (GBT), and Multi-Layered Perceptron (MLP). For each model type, the Scikit-Learn Python implementation was employed [22].

C. Performance metrics and Model Selection

Traditionally, the ML community has used prediction accuracy and error rate measures as simple performance metrics. However, neither accuracy nor error rate can provide adequate information about classifier performance in imbalanced datasets. Because of class imbalance, many SE studies have shifted to using alternative performance metrics such as the Area Under the Curve (AUC) [8], [23].

The selection criterion used to identify the best model was 5-fold cross validation with the AUC metric. Here by selecting a model, we mean making the choices of:

1) preprocessing hyperparameters
   - SMOTE (`True` or `False`)
   - log transform (`True` or `False`)
2) model-type (LR, RF, GBT, MLP)
3) hyperparameters specific to the model type (e.g., L2 regularisation for LR, number of trees for RF etc.)

A grid search was conducted to explore each combination of model type and methodological hyperparameters. In contrast, a random search was used to examine the sets of hyperparameters specific to each model type employed. We are left with one model per dataset, which are evaluated on the held-out test sets to estimate generalization performance. These are the models whose prediction (on the test set) we aim to explain with LIME and SHAP.

D. Explanations and Disagreements

LIME and SHAP are both model agnostic, meaning that one can apply them to explain the decisions of any of our models. However, this versatility does not come freely: the feature attributes provided by LIME and SHAP need not agree on every single test instance we explain. This disagreement issue between post-hoc explainers has already been observed, and various agreement metrics were recently introduced to characterize the different degrees of disagreements [10]. In this work, we center around the accompanying three metrics.

- **Feature Agreement (FA)** computes the fraction of common features between the sets of top-k features of two explanations.
- **Rank Agreement (RA)** computes the fraction of features that are common between the sets of top-k features of two explanations **and** that have the same position in the respective rank orders.
- **Sign Agreement (SA)** computes the fraction of features that are common between the sets of top-k features of two explanations, **and** share the same sign (positive/negative) in both explanations.

Formally, if we let E_a and E_b be two explanations, top(E_a, k) be the set of top-k features in E_a, rank(E_a, i) be the rank of

feature i in E_a, and sign(E_a, i) be the sign (positive/negative) of feature i in the explanation E_a, we can define the set

$$S = \text{top}(E_a, k) \cap \text{top}(E_b, k) \quad (1)$$

representing the most important features according to **both** LIME and SHAP. The feature agreement defined as

$$\text{FA}(E_a, E_b) = \frac{|S|}{k} \quad (2)$$

where $|S|$ denotes the cardinality of the set S. Moreover the rank agreement and sign agreement are

$$\text{RA}(E_a, E_b) = \frac{|\{i \in S : \text{rank}(E_a, i) = \text{rank}(E_b, i)\}|}{k} \quad (3)$$

and

$$\text{SA}(E_a, E_b) = \frac{|\{i \in S : \text{sign}(E_a, i) = \text{sign}(E_b, i)\}|}{k} \quad (4)$$

respectively. The three metrics FA, RA, and SA take values between 0 (complete disagreement) and 1 (complete agreement). Hence, empirically observing a significant number of LIME and SHAP explanations with low values of these metrics is evidence of disagreements between explanations.

As an illustrative example with $k = 3$, we go back to Figure 3. On the left example, the top-3 features of LIME and SHAP only have two features in common i.e., FA $= 2/3$. The sign of the importance of these features are consistent between LIME and SHAP and therefore the sign agreement is the same as the feature agreement. Finally, out of the two features common in both top-3s, only `OWN_COMMIT` has a consistent rank leading to RA $= 1/3$. In the example on the right, we have perfect feature agreement, i.e., FA $= 1$. Nonetheless, the sign of the importance of `OWN_COMMIT` is not the same between LIME and SHAP which means that SA $= 2/3$. Moreover, none of the features common to the top-3 of both explainers have the same rank so we have RA $= 0$.

For the experiments, the value of k was different depending on the number of features used in the best model for each dataset. The parameter k was set to 10 when 24 or more features were used, and set to 5 otherwise.

IV. Results

Table I presents the AUC performance measures for the models that were selected on every single dataset. We first observe that Random Forests are the dominating model seeing they were chosen on 9 out of the ten datasets. Secondly, we can compare the cross-validated AUC to the test set AUC and observe that they are similar, meaning that 5-fold cross-validation provided an accurate estimate of generalization performance in our experiments. Moreover, all test-AUCs are above 0.75 which prior work advocates is the minimal value required to have reliability of the interpretations [23].

Satisfied with the performance of the models, we ran LIME and SHAP explanations on files which were predicted as defective. The three disagreements metrics between both explanation techniques were then computed on all these modules and the distribution of disagreements is shown in Figure 4.

TABLE I
Performance of Selected Models

Dataset	Model	CV-AUC	test-AUC
activemq-5.8.0	RF	0.84	0.84
groovy-1_6_BETA_2	RF	0.88	0.88
hive-0.10.0	RF	0.88	0.85
hive-0.12.0	RF	0.88	0.88
hive-0.9.0	RF	0.89	0.92
lucene-2.3.0	RF	0.93	0.94
lucene-2.9.0	RF	0.82	0.80
lucene-3.0.0	RF	0.85	0.83
wicket-1.3.0-beta2	RF	0.85	0.81
wicket-1.5.3	LR	0.81	0.78

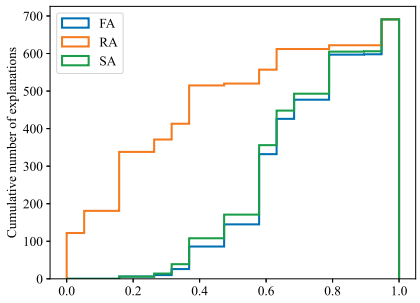


Fig. 4: Cumulative Distribution Function (CDF) of the FA, RA, and SA agreement measures of LIME and SHAP explanations computed for all files predicted as defective across the ten datasets.

We observe a left-ward shift of the distributions of rank agreements compared to sign agreements, suggesting that there are more disagreements regarding the ranking of importance features than the sign of their attributions.

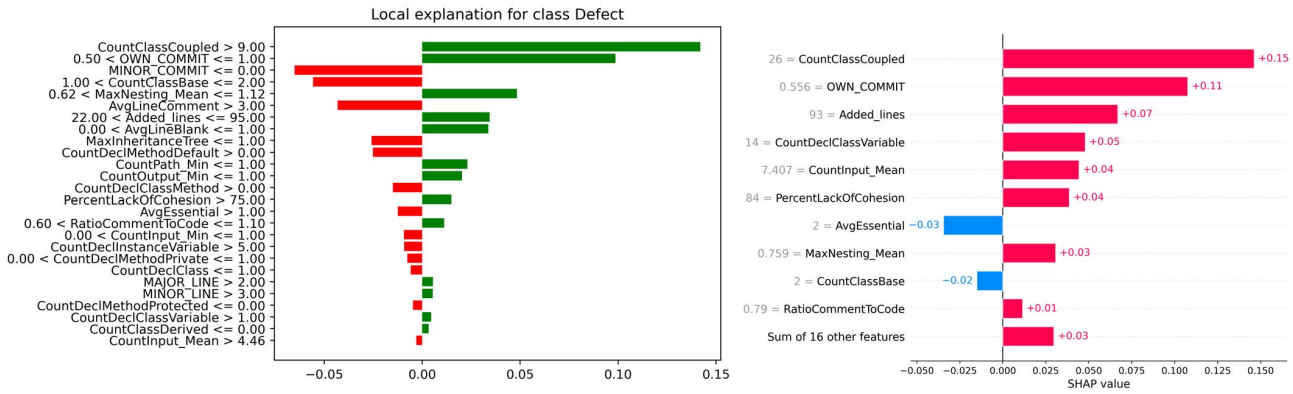
Seeing that distribution of disagreements between FA and SA are very close, we deduce that when features are common to the top-k of both explainers, then the sign of their attributions tend to agree. We cannot say the same thing regarding the ranks of these features. This observation suggests that to reduce the perceived disagreement betweem the two explainers, one should ignore ranks altogheter and study the Aggregated set

$$A = \{i \in S : \text{sign}(E_a, i) = \text{sign}(E_b, i)\} \quad (5)$$

at the numerator of Equation 4 Features in A can be given a color: green if both LIME and SHAP agree that this feature's importance is positive, and red if both LIME and SHAP agree that this feature's attribution is negative. The advantage of this approach is that information on which LIME and SHAP disagree is not shown to the user, which can potentially reduced the "perceived" disagreements. However, rigorous statements about perception of disagreements remain to be made via a user study, which we leave as a future work.

V. Conclusion and Future Works

Defect prediction is a procedure where ML models predict software defects based on data collected from the repositories. Still, most ML models are black boxes and it is difficult to extract human understandable reasoning behind their decisions.



Common top-10 with same sign

A = {CountClassCoupled, OWN_COMMIT, Added_lines, CountClassBase, MaxNesting_Mean}

Fig. 5: Example of how the proposed aggregation mechanism works. We only present the user with the set A representing features common in the top-10 of both LIME and SHAP, and whose importance has the same sign.

As an attempt to solve this inherent limitation, there is a growing interest in employing post-hoc explanations methods such as LIME and SHAP to get insight into the reasoning of a model. In this study, we show however that LIME and SHAP cannot be used interchangeably considering that their explanations show disagreements across multiple defect prediction datasets. Also, we show that LIME and SHAP tend to disagree more regarding the ranking of important features rather than the sign of their importance (positive or negative). This observation has led us to propose an aggregation scheme to combine LIME and SHAP explanations in a manner that accentuates their agreements, and hides their disagreements.

In the future, we plan to conduct a user study to see – (1) how practitioners deal with disagreements between post-hoc explainers and (2) whether our aggregation mechanism genuinely reduces the perception of disagreement between explanations.

References

[1] Z. Li, X.-Y. Jing, and X. Zhu, "Progress on approaches to software defect prediction," *Iet Software*, vol. 12, no. 3, pp. 161–175, 2018.

[2] S. K. Das, S. Roy, D. Sarker, and M. M. Rahman, "Energy efficient failure recovery with concurrent execution in hadoop cluster," 2016.

[3] M. K. Thota, F. H. Shajin, P. Rajesh *et al.*, "Survey on software defect prediction techniques," *IJASE*, vol. 17, no. 4, pp. 331–344, 2020.

[4] S. Strüder, M. Mukelabai, D. Strüber, and T. Berger, "Feature-oriented defect prediction," in *Proc. SPLC*, 2020.

[5] M. S. Rawat and S. K. Dubey, "Software defect prediction models for quality improvement: a literature study," *IJCSI*, vol. 9, no. 5, p. 288, 2012.

[6] M. T. Ribeiro, S. Singh, and C. Guestrin, "Why should i trust you?" explaining the predictions of any classifier," in *Proc. SIGKDD*, 2016.

[7] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Proc. NIPS*, 2017.

[8] J. Jiarpakdee, C. Tantithamthavorn, H. K. Dam, and J. Grundy, "An empirical study of model-agnostic techniques for defect prediction models," *TSE*, 2020.

[9] G. K. Rajbahadur, S. Wang, G. Ansaldi, Y. Kamei, and A. E. Hassan, "The impact of feature importance methods on the interpretation of defect classifiers," *TSE*, 2021.

[10] S. Krishna, T. Han, A. Gu, J. Pombra, S. Jabbari, S. Wu, and H. Lakkaraju, "The disagreement problem in explainable machine learning: A practitioner's perspective," *arXiv preprint arXiv:2202.01602*, 2022.

[11] C. Lewis, Z. Lin, C. Sadowski, X. Zhu, R. Ou, and E. J. Whitehead, "Does bug prediction support human developers? findings from a google case study," in *Proc. ICSE*, 2013.

[12] C. Pornprasit, C. Tantithamthavorn, J. Jiarpakdee, M. Fu, and P. Thongtanunam, "Pyexplainer: Explaining the predictions of just-in-time defect models," in *Proc. ASE*, 2021.

[13] J. Jiarpakdee, C. K. Tantithamthavorn, and J. Grundy, "Practitioners' perceptions of the goals and visual explanations of defect prediction models," in *Proc. MSR*, 2021.

[14] M. Staniak and P. Biecek, "Explanations of model predictions with live and breakdown packages," *arXiv preprint arXiv:1804.01955*, 2018.

[15] K. Peng and T. Menzies, "Defect reduction planning (using timelime)," *TSE*, vol. 48, no. 7, pp. 2510–2525, 2022.

[16] J. Shin, R. Aleithan, J. Nam, J. Wang, and S. Wang, "Explainable software defect prediction: Are we there yet?" *arXiv preprint arXiv:2111.10901*, 2021.

[17] S. Yatish, J. Jiarpakdee, P. Thongtanunam, and C. Tantithamthavorn, "Mining software defects: should we consider affected releases?" in *Proc. ICSE*, 2019.

[18] J. Jiarpakdee, C. Tantithamthavorn, and C. Treude, "Autospearman: Automatically mitigating correlated metrics for interpreting defect models," *arXiv preprint arXiv:1806.09791*, 2018.

[19] 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.

[20] "Types of classification algorithms in machine learning." [Online]. Available: https://bit.ly/3OLPmt2

[21] F. Calefato, F. Lanubile, and N. Novielli, "An empirical assessment of best-answer prediction models in technical q&a sites," *Empirical Software Engineering*, vol. 24, no. 2, pp. 854–901, 2019.

[22] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[23] Y. Lyu, G. K. Rajbahadur, D. Lin, B. Chen, and Z. M. Jiang, "Towards a consistent interpretation of aiops models," *TOSEM*, vol. 31, no. 1, pp. 1–38, 2021.

View publication stats