Explainable AI for Issue Classification: A Multi-class Study with LIME and SHAP

Jueun Heo
Department of AI Convergence Engineering
Gyeongsang National University
Jinju, Republic of Korea
juandeun@gnu.ac.kr

Seonah Lee
Department of AI Convergence Engineering
Department of Software Engineering
Gyeongsang National University
Jinju, Republic of Korea
saleese@gnu.ac.kr

***Abstract*—Issue classification is a fundamental task in software development, enabling teams to manage issue reports. Automatic issue classification can help developers classify issue reports. However, developers should understand why each issue report is classified in such a way. A prior study has shown that explainable AI (XAI) can explain how an issue report is classified as a bug or a non-bug. However, the binary setting limits applicability to real-world issue tracking systems, where multiple categories coexist. In this paper, we replicate and extend the prior study by conducting a multi-class issue classification experiment using three categories: Bug, Enhancement, and Question. We use a fine-tuned, seBERT-based classifier and apply two widely used XAI models, LIME and SHAP, to generate explanations for issue classification. We then analyze the results of applying LIME and SHAP to multi-class issue classification, both qualitatively and quantitatively.**

***Index Terms*—Issue, Classification, Replication Study, LIME, SHAP, Explainable AI**

I. Introduction

Issue management effectively plays a critical role in reporting, categorization, and management of software issues. However, due to the high volume and diverse nature of issue reports, it is time-consuming for developers to manually classify issue reports. To address this issue, researchers have proposed automating the classification of issue reports using deep learning and machine learning techniques [1]–[16].

However, since automatic issue classification is not always correct, developers' judgment remains necessary. Therefore, it is important that developers can refer to the underlying rationale of automatic issue classification. To address this, Schulte et al. [17] proposed the interpretability of a binary issue classification (Bug vs. Non-bug) using explainable AI (XAI) methods such as LIME and SHAP. While valuable, the limitation of focusing only on binary classification reduces the applicability of such findings to real-world settings, where issue reports typically span multiple categories.

In this paper, we replicate and extend the prior study by moving from binary to multi-class issue classification. Specifically, we investigate three common categories — Bug, Enhancement, and Question — and evaluate the explanations generated by LIME and SHAP in this multi-class setting. Our study aims to answer the following research questions:

RQ1: How do explanation qualities differ in multi-class issue classification compared to binary classification?

RQ2: Are certain classes (e.g., Bug, Enhancement, and Question) inherently easier to explain than others?

RQ3: In the presence of misclassified issues, can their explainability contribute to user understanding?

By addressing these questions, we provide insights into the applicability of XAI for more realistic and fine-grained issue classification scenarios.

II. Methodology

In this section, we explain the methods for applying XAI to multi-class classification tasks and for analyzing the application results.

A. seBERT-based Multi-class Issue Classification

In this study, we used an seBERT-based multi-class classification model. In particular, we adopted the seBERT-based three-class issue classification model that had been fine-tuned and made publicly available in previous research [18][1]. The model is based on seBERT [19] as its backbone, and the model performs multi-class classification into one of 'bug', 'enhancement', or 'question' through a softmax output when given the concatenated title and body of an issue as input.

B. Multi-class Issue Dataset

We used the test set published by Trautsch et al. [18]. The dataset contains GitHub issue titles and bodies, with labels of 'bug', 'enhancement', or 'question'. Trautsch et al. [18] mainly evaluated predictive performance, such as classification accuracy, on this dataset. In contrast to Trautsch et al.'s work, our study performs explainability (XAI) analysis with LIME [20] and SHAP [21] on the same test set, and qualitatively and quantitatively assesses the validity of the highlighted rationales.

C. Explainability Methods: LIME and SHAP

In issue classification, the LIME(Local Interpretable Model-agnostic Explanations) model [20] explains which words influenced the prediction made for a given issue report. The mechanism is as follows. First, the words of the target report are transformed into an interpretable representation. For example, the issue report can be converted into a vector indicating

[1]https://github.com/atrautsch/nlbse2022_replication_kit

2151-0849/25/$31.00 ©2025 IEEE
DOI 10.1109/ASEW67777.2025.00056

whether each word is present (1) or absent (0). Next, several perturbed samples are generated by randomly removing some words from the original report, and these samples are fed into a classification model to obtain prediction probabilities. Then, a similarity function is used to assign weights based on how close each sample is to the original report, and a sparse linear model is trained on this weighted data. This linear model approximates the behavior of the classification model in the local region around the issue report, ultimately showing the contribution of each word to the prediction in the form of weights. For instance, if a particular issue is classified as a 'bug,' LIME may visually explain that words such as 'does,' 'not,' and 'work' contributed positively to the prediction, while words like 'docs' served as counter-evidence.

SHAP (SHapley Additive exPlanations) [21] is a model for quantitatively estimating the contribution of each word to the prediction of a text classification model. First, a sentence is transformed into a vector representation such as Bag-of-Words or embeddings. After that, the trained classification model computes a prediction for the input. SHAP defines a word's contribution as the average change in the model's output, when the word is included versus excluded. For that, SHAP computes all possible combinations of words. Starting from a base value (the average prediction when no word information is available), SHAP reconstructs the final prediction by sequentially adding the SHAP values assigned to individual words. Thus, SHAP not only identifies which words are important, but also quantitatively shows "to what extent this word shifts the prediction toward a positive or negative outcome." For example, when an issue is classified as a bug, SHAP can transparently explain the model's decision by indicating numerically that the word 'not' contributed positively to the prediction.

D. Qualitative Analysis

To answer three research questions, we sampled three cases that are correctly classified and other three cases that are incorrectly classified. We then visualized these cases with the explanation results generated by LIME and SHAP.

For each case, we visualized the top words highlighted by LIME and SHAP. We then manually examined the semantic relations of the highlighted words to the classified ones (e.g., Bug, Enhancement, and Question). Based on the manual examination, we determined whether the highlighted words are aligned with the classified ones.

Additionally, for misclassified cases, we also manually investigated individual cases to assess whether highlighted words help user understanding or cause user confusion. By doing so, we qualitatively evaluated how well the explanation given by LIME and SHAP are aligned with the actual labels.

E. Quantitative Analysis

To support our qualitative analysis, we also conducted an quantitative analysis of XAI for our multi-issue classification. For the analysis, we applied the XAI models to the classification of 300 issue data.

We followed Lu et al. [22] and measured the faithfulness of XAI models. Here, faithfulness refers to the degree to which the important input features actually used by the model for prediction are accurately reflected in the explanations generated LIME or SHAP. In other words, it is an indicator of how faithfully the explanation follows the model's true decision-making rationale.

To measure the faithfulness of XAI models, we used three metrics: Log-Odds, Comprehensiveness and Sufficiency. The meaning of each metric is as follows.

- **Log-Odds** [23]: The metrics measure how the model's confidence changes in log-odds space when the top-k important words are removed or retained. Drop in Log-Odds quantifies the decrease in model confidence when the important words are removed. Keep in Log-Odds quantifies how well the model's confidence is preserved when only the important words are retained. A large decrease (more negative value) in Drop and a value close to zero in Keep indicate high faithfulness.
- **Comprehensiveness** [24]: The metric measures how much the prediction probability decreases when the top-k words are removed. A higher value indicates that the removal of important words has a significant impact on the model's prediction, suggesting that the explanation is comprehensive.
- **Sufficiency** [24]: The metric measures how well the prediction is preserved when only the top-k words identified by the explanation method are retained and the others are removed. A lower value indicates that the prediction can be made with only a few words, suggesting higher faithfulness.

We computed each metric for each issue classification. These metrics provide a multifaceted evaluation of the faithfulness of XAI models.

III. Qualitative Analysis Results

To answer our research questions, we manually examined the issue classification results and their explainability. As shown in Figure 1 and Figure 2, we were able to observe the LIME and SHAP results for issue reports classified as bug, enhancement, and question. Based on these results, we discuss the following three research questions. The experimental code and results are available at the following repository.[2].

A. RQ1: How do explanation qualities differ in multi-class issue classification compared to binary classification?

1) Qualitative Analysis: We found that the quality of explanations for multi-class issue classification is comparable to that of binary classification. For example, in the case of LIME as shown in Figure 1(a), even when there are three classes—bug, enhancement, and question—if an issue is classified as a bug, the words contributing to the bug classification are highlighted in blue. Likewise, in the case of SHAP, the words contributing to the bug classification are highlighted in red. Both methods

[2]https://github.com/jueun4136/XAI-Issue-Classification

demonstrate that the issue was classified as a bug, because the word 'crashed' appeared in the issue report.

> The quality of explanations for multi-class issue classification is comparable to that of binary classification, as words are distinguished between those contributing to the assigned category and those that do not.

B. RQ2: Are certain classes (e.g., Bug, Enhancement, and Question) inherently easier to explain than others?

We observed that the explanations appear to be easiest in the order of Bug, Enhancement, and Question. As illustrated in Figures 1(a) and 1(b), bugs can be directly associated with words such as 'incompatibility' and 'crashed,' which clearly indicate the presence of a problem. Figures 1(b) and 1(c) provide examples for enhancement, where words like 'add' and 'creation' show some degree of relevance to improvements, but their connection is not as direct. Finally, Figures 1(c) and 1(d) demonstrate the case of questions, where the word 'envoy' was selected as a contributing term. While the question mark symbol '?' could potentially contribute to classification, it appears to have been excluded since the analysis is word-based.

> The ease of explanation differs across classes. Where LIME and SHAP reveal associations between words and classifications, some classes are easily associated with words, while others are not.

C. RQ3: In the presence of misclassified issues, can their explainability contribute to user understanding?

To support developers' justification, the explanation for automatic issue classification should help their understanding, even in cases of mis-classification. To the end, we identified misclassified cases and examined the results of LIME and SHAP. Figure 2 presents illustrative examples. In Figure 2(a), the model classified the issue as 'question' with a probability of 0.57. However, it also highlights that words such as 'error', which are related to the true class 'bug' but are not associated with 'question.' Therefore, we observed that the LIME result helps developers' understanding. In contrast, SHAP in Figure 2(b) designates 'error' as a word related to 'question,' which hinders developers' understanding.

Next, Figure 2(c) illustrates a case where the issue was classified as 'bug' but actually belongs to the 'enhancement' class. The issue was classified as 'bug' with a probability of 0.97, and words such as 'fix' and 'issue' are evidently associated with the 'bug' class, even from a human perspective. Therefore, this case can be regarded as one where the issue report itself is inherently difficult to be classified as 'enhancement.' Figure 2(d) with SHAP also shows that the classification as Bug was driven by words like 'fix' and 'issue.'

Finally, Figure 2(e) shows a case where the issue was classified as 'enhancement' but is actually a 'question.' In this case, the classification as 'enhancement' was influenced by words such as 'allow'; however, words like 'question,' which are not related to 'enhancement,' were also highlighted. Thus, when developers review the LIME results, they may be able to correctly interpret the issue as a 'question.' Similarly, Figure 2(f) with SHAP also identifies terms such as question as unrelated to Enhancement, thereby supporting developers' understanding.

> In cases of misclassified issues, explainability of issues can aid user understanding, but not always.

IV. Quantitative Analysis Results

We have addressed our research questions through qualitative evaluation. However, to deepen the understanding of our XAI models, we also provide additional quantitative results such as the classification accuracy of the models we examined, faithfulness of XAI models, etc.

A. Classification Accuracy

We first selected 100 issue samples for each of the bug, enhancement, and question categories and performed multi-issue classification. Table I shows the results.

TABLE I: Classification Accuracy per Each Class

Label	Precision	Recall	F1-score	Support	TP
bug	0.71	0.94	0.81	100	94
enhancement	0.75	0.85	0.79	100	85
question	0.96	0.51	0.67	100	51
Macro avg	0.80	0.77	0.76	300	-
Weighted avg	0.80	0.77	0.76	300	-

The classification yielded an average precision of 0.80, recall of 0.77, and F1-score of 0.76. The classification accuracy was a little bit lower than that reported in previous work [18]. This result may be attributed to the reduced number of question samples, which were originally more abundant in the dataset but were limited to 100 in our experimental setup.

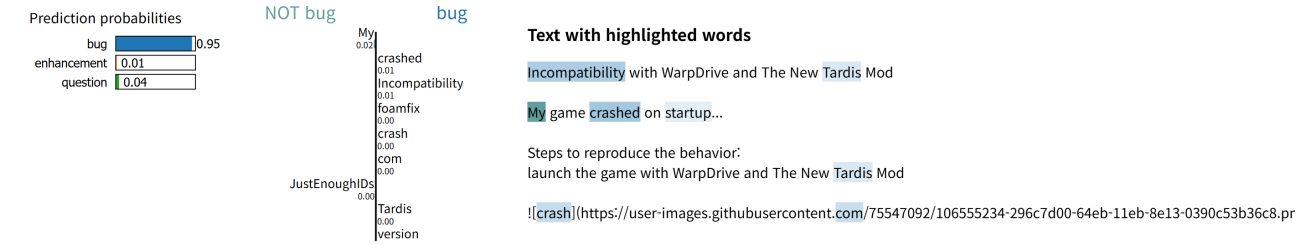
B. Highly Ranked Tokens per Each Class

We applied SHAP and LIME to the 300 classified issue data and collected the highlighted words for each category. As a result, we identified the top 15 tokens highlighted for each class. Figure 3 presents the results. Among the top 15 words, those containing non-English characters or single numeric digits were removed during post-processing.

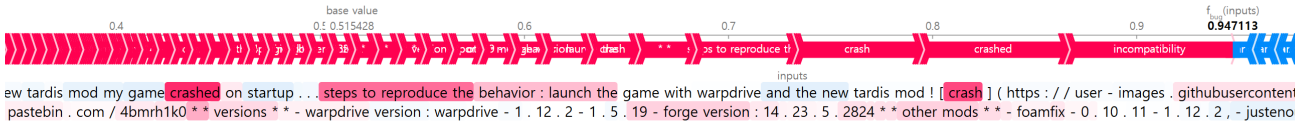
The left figure of Figure 3(a) shows the tokens that contribute most to explaining issues classified as Bug, when SHAP is applied. The top tokens include 'not', followed by 'images' and 'Describe the.' The token, 'images,' suggests that bug reports may often include image files, implying a potential link between image attachments and bug-related issues.

The left figure of Figure 3(b) shows the tokens that contribute most to explaining issues classified as Bug, when LIME is applied. The top tokens include 'bug', followed by 'not' and 'reproduce.' These tokens appear to be highly relevant to the Bug category.

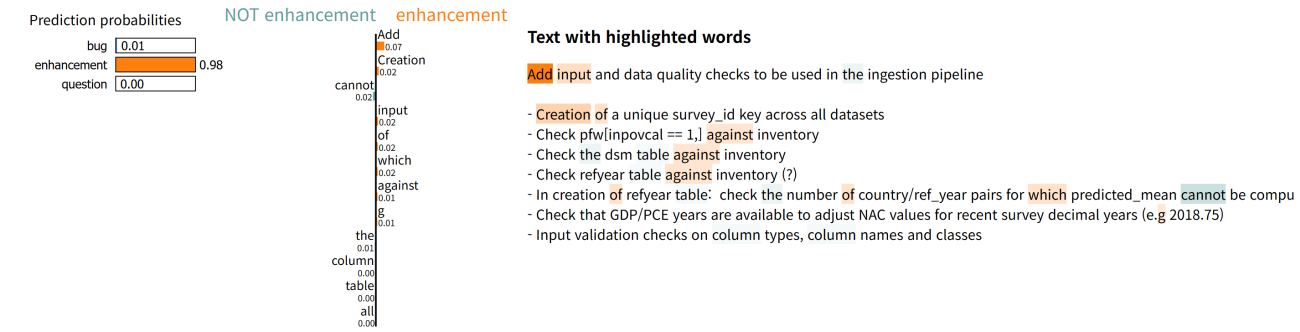
The center figure of Figure 3(a) shows the tokens that contribute to explaining issues classified as Enhancement,



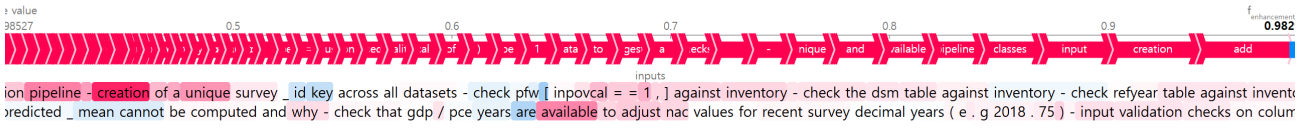
(a) LIME (case 04, classified as Bug)



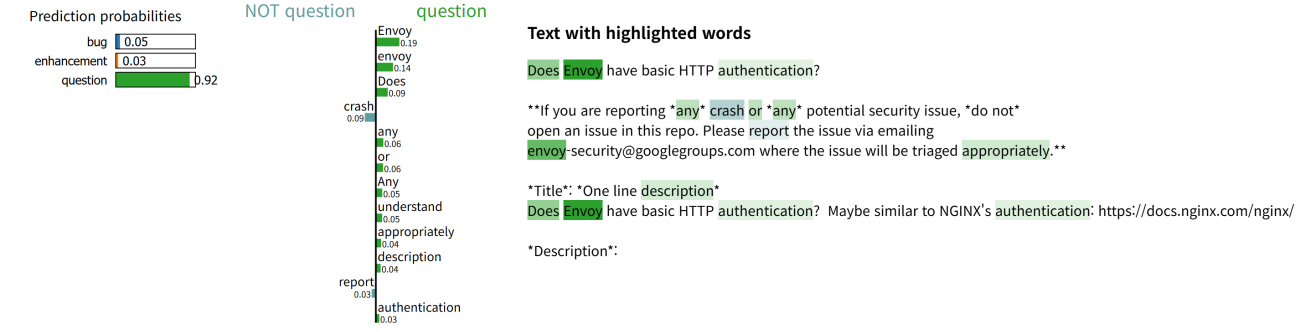
(b) SHAP (case 04, classified as Bug)



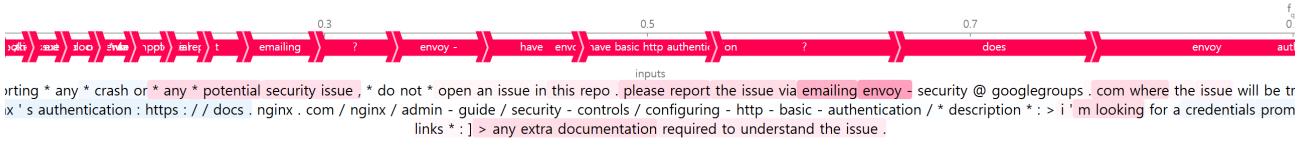
(c) LIME (case 10, classified as Enhancement)



(d) SHAP (case 10, classified as Enhancement)

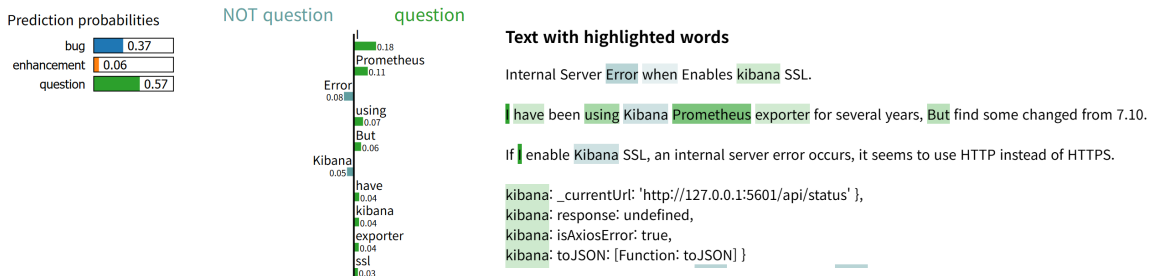


(e) LIME (case 28, classified as Question)

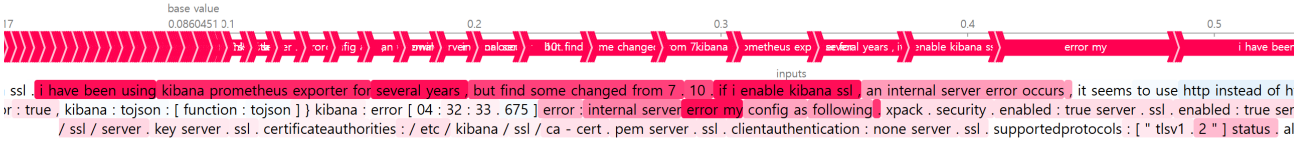


(f) SHAP (case 28, classified as Question)

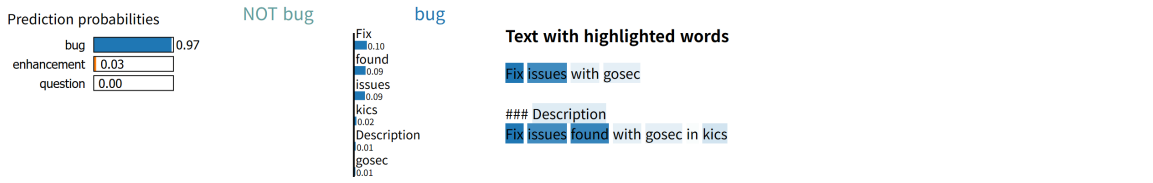
Fig. 1: Case studies of explaining automatic issue classification (LIME vs. SHAP)



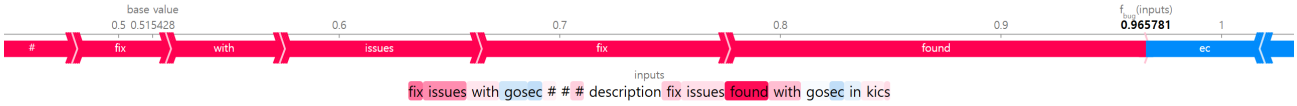
(a) LIME (case 08, classified as Question, but should be classified Bug)



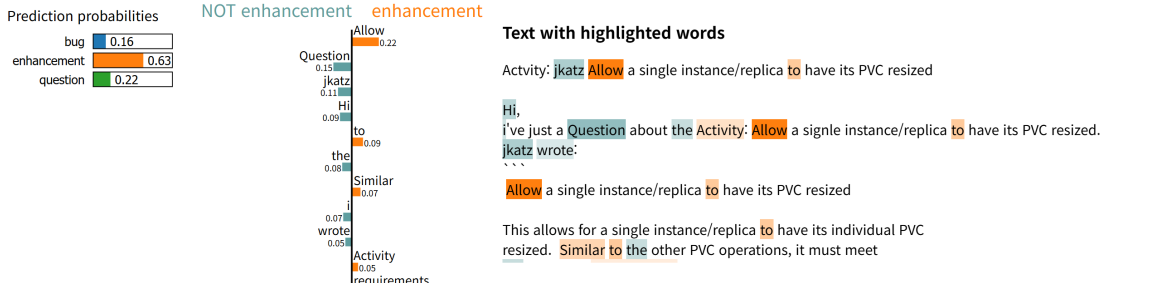
(b) SHAP (case 08, classified as Question, but should be classified as Bug)



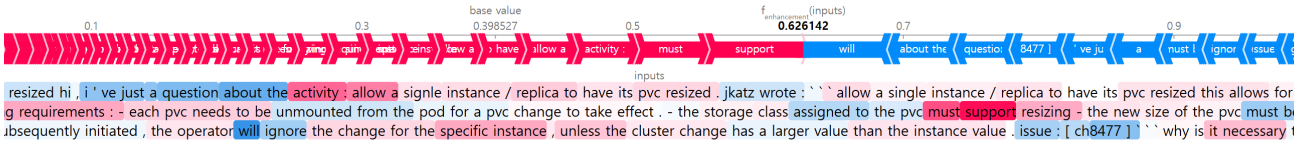
(c) LIME (case 18, classified as Bug, but should be classified as Enhancement)



(d) SHAP (case 18, classified as Bug, but should be classified as Enhancement)

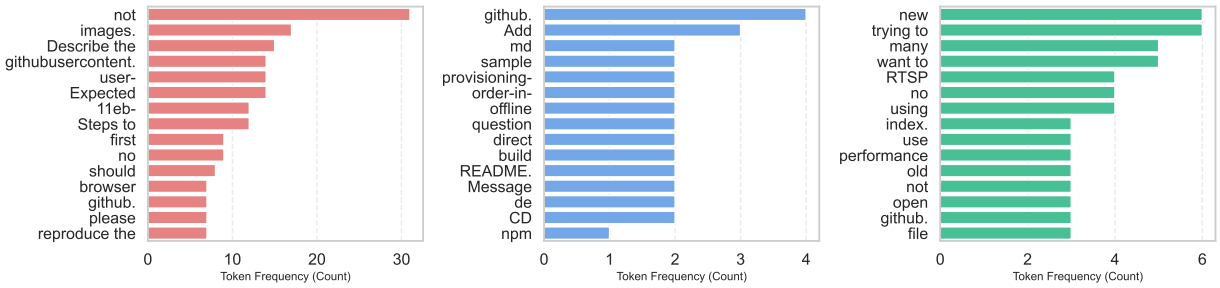


(e) LIME (case 23, classified as Enhancement, but should be classified as Question)

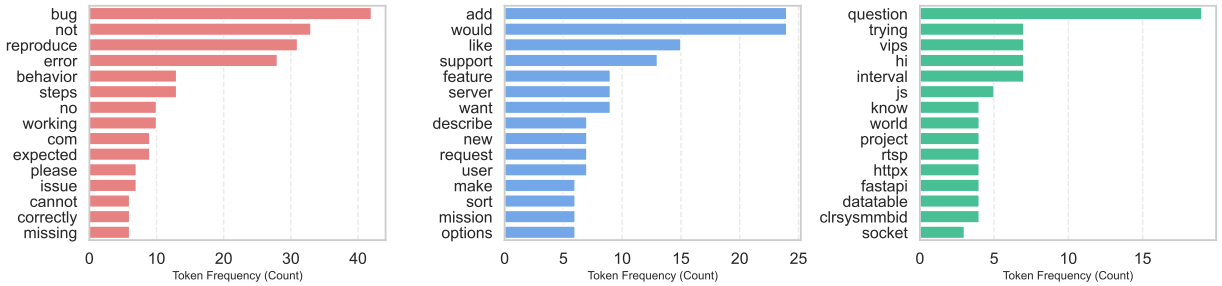


(f) SHAP (case 23, classified as Enhancement, but should be classified Question)

Fig. 2: Case studies of explaining incorrect automatic issue classification (LIME vs. SHAP)



(a) Top 15 tokens in the explanations of SHAP



(b) Top 15 tokens in the explanations of LIME

Fig. 3: Top 15 tokens in the explanations of XAI models

when SHAP is applied. The top-ranked token is 'github', followed by the verb 'Add' and then 'md' and 'sample.'

The center figure of Figure 3(b) shows the tokens that contribute to explaining issues classified as Enhancement, when LIME is applied. The top-ranked token is 'add', followed by the modal verb 'would' and then 'like' and 'support.' Through the analysis, we observe that the word 'add' contributes to the classification into the Enhancement category.

The right figure of Figure 3(a) shows the tokens that explain issues classified as Question, when SHAP is applied. In this case, the top-ranked token is 'new,' followed by 'trying to,' and then 'many.' The right figure of Figure 3(b) shows the tokens that explain issues classified as Question, when LIME is applied. In this case, the top-ranked token is 'question,' followed by 'trying,' and then 'vips.'

When examining the highlighted words contributing to the explanations of SHAP and LIME, we found that LIME provides more intuitively interpretable terms for each category, compared to SHAP.

C. Faithfulness of SHAP and LIME

We discuss the faithfulness of SHAP and LIME. Faithfulness evaluation was conducted only on correctly classified samples (Table II, column "TP").

1) Log-Odds: Table II presents the mean faithfulness scores comparing LIME and SHAP across different top-k feature selections. The results demonstrate that LIME consistently outperforms SHAP across all metrics and k values.

In the case of Log-Odds Drop, LIME consistently showed larger negative values than SHAP across all values of k. In particular, it exhibited the strongest effect at k=20 with a score of -0.5771. This indicates that removing the important features identified by LIME significantly reduces the model's confidence. For example, at k=5, LIME's score (-0.3299) is considerably lower than SHAP's score (-0.0758), suggesting that the features selected by SHAP had a relatively weaker impact on the model's prediction.

In the case of Log-Odds Keep, the values tended to be close to zero, as the value of k increased. This indicates that as K becomes larger, sufficient information is retained to reproduce the prediction. Across all values of k, LIME's values consistently remained closer to zero than those of SHAP, suggesting that LIME more faithfully explains the model's decision-making. Therefore, in terms of Log-Odds Keep, LIME maintained higher faithfulness than SHAP.

2) Comprehensiveness and Sufficiency: Figure 4 presents comparisons of SHAP and LIME in terms of the faithfulness metrics—Comprehensiveness and Sufficiency. Here, k denotes the top-k tokens that most strongly influenced the model's prediction in each explanation. We conducted experiments for k values of 5, 10, and 20.

The upper part of Figure 4 presents a comparison between SHAP and LIME based on the Comprehensiveness metric. Comprehensiveness assesses whether the identified tokens are truly important. When the top-k tokens—deemed most influential to the model's prediction—are removed from the input,

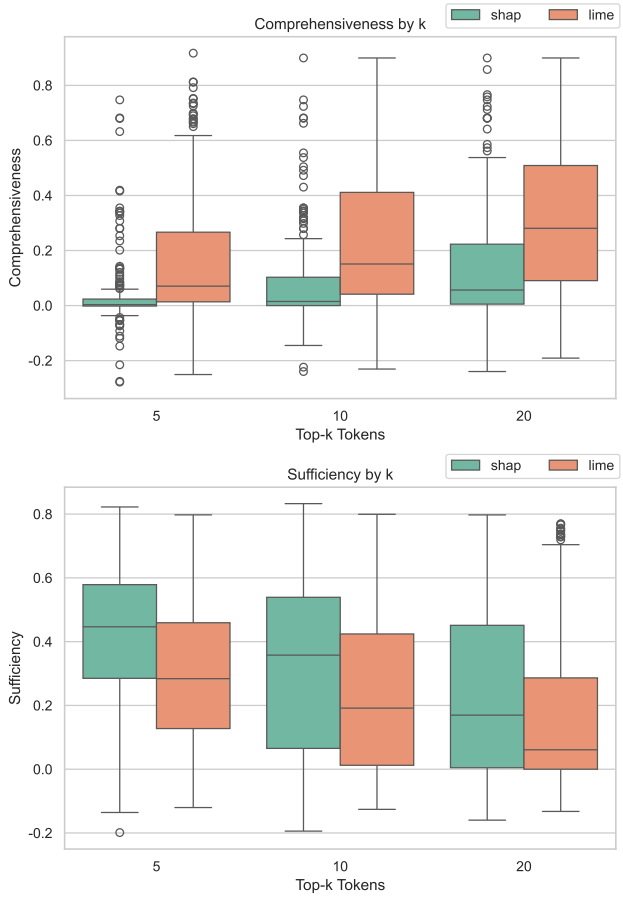


Fig. 4: comprehensiveness and sufficiency by top-k tokens

the extent to which the model's output probability decreases is measured. A significant drop in the predicted probability indicates that the removed tokens were indeed important to the model's decision. Higher Comprehensiveness values indicate better faithfulness.

The experimental result shows that for all values of k (5, 10, 20), the median scores of LIME are higher than those of SHAP. This indicates that removing the top tokens identified by LIME leads to a greater drop in the model's prediction confidence, suggesting that LIME captures tokens more critical to the model's decision-making.

The lower part of Figure 4 presents a comparison between SHAP and LIME based on the Sufficiency metric. Sufficiency assesses whether the identified tokens alone are sufficient for the model's prediction. When only the top-k tokens—those deemed most important—are kept and all other tokens are removed, how much of the model's output probability is retained is measured. If the predicted probability remains close to the original, it suggests that the selected tokens provide a sufficiently explanatory basis for the prediction. Lower Sufficiency values indicate better faithfulness.

The experimental results show that LIME consistently achieves lower median values than SHAP. It indicates that the model's prediction confidence is better preserved when only the top tokens identified by LIME are retained. This suggests that LIME captures a more essential and sufficient set of tokens for the model's prediction.

k	Method	Log Odds		Metrics	
		Drop (↓)	*Keep (→0)*	*Comp. (↑)*	*Suff. (↓)*
5	LIME	**-0.3299**	**-0.5161**	0.1621	0.2811
	SHAP	-0.0758	-0.7168	0.0395	0.3766
10	LIME	**-0.4729**	**-0.4212**	0.2274	0.2271
	SHAP	-0.1436	-0.5834	0.0766	0.3056
20	LIME	**-0.5771**	**-0.2905**	0.2885	0.1568
	SHAP	-0.2424	-0.4177	0.1304	0.2192
FULL	LIME	**-0.4600**	**-0.4093**	0.2260	0.2217
	SHAP	-0.1539	-0.5726	0.0822	0.3005

TABLE II: Mean faithfulness scores of LIME and SHAP. Arrows (↑/↓) indicate higher or lower is better; Keep (→0) indicates closer to zero is better.

3) Faithfulness of Three Categories: Figure 5 analyzes the faithfulness of explanations given by LIME and SHAP for each category—Bug, Enhancement, and Question—from the perspectives of Comprehensiveness and Sufficiency. As lower Sufficiency and higher Comprehensiveness indicate better faithfulness, the Enhancement category demonstrates the highest faithfulness, followed by Bug, and finally Question. In addition, consistent with the previous analyses, LIME shows higher faithfulness than SHAP.

V. Discussion

We discuss the implications and limitations of our study.

A. Differences in Explanation Quality in Multi-class Issue Classification

Our findings indicate that extending explainability from binary to multi-class issue classification does not necessarily degrade explanation quality. Similar to the binary setting, both LIME and SHAP are able to distinguish words contributing to the assigned class from those that do not.

B. Variation in Explanation Difficulty Across Classes

The ease of explanation is not uniform across classes. Certain categories, such as Bug, are relatively easier to explain because they are often associated with concrete and domain-specific words (e.g., error, fix, issue). By contrast, other categories such as Question are harder to explain, as their linguistic signals are less distinctive and more context-dependent.

C. Limitations

This study has several limitations. Our research questions were mainly assessed qualitatively, which leaves room for bias and restricts statistical confidence in the findings. Moreover, our work focuses on multi-class issue classification; it remains unclear how explainability would behave in a multi-label setting, where an issue may belong to multiple categories simultaneously.

Fig. 5: Comprehensiveness–Sufficiency distribution of LIME (orange) and SHAP (green) for top-k = 10 across the Bug, Enhancement, and Question classes. Points concentrated toward lower Sufficiency (↓) and higher Comprehensiveness (↑) indicate higher faithfulness.

VI. Conclusion

In this paper, we replicated a prior study on explainable AI for issue classification and extended it from binary (Bug vs. Non-bug) to a multi-class setting involving Bug, Enhancement, and Question. Our results indicate that multi-class classification is feasible with a seBERT-based model, but explanation quality varies across categories, sensitive to class characteristics and error cases. For future work, we plan to expand the set of issue categories, incorporate larger datasets, and conduct user studies with developers to evaluate the usefulness of explanations in real project settings.

References

[1] Q. Fan, Y. Yu, G. Yin, T. Wang, and H. Wang, "Where is the road for issue reports classification based on text mining?," in *2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, pp. 121–130, IEEE, 2017.

[2] R. Kallis, A. Di Sorbo, G. Canfora, and S. Panichella, "Ticket tagger: Machine learning driven issue classification," in *2019 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pp. 406–409, IEEE, 2019.

[3] J. Heo, G. Kwon, C. Kwak, and S. Lee, "A comparison of pretrained models for classifying issue reports," *IEEE Access*, 2024.

[4] G. Aracena, K. Luster, F. Santos, I. Steinmacher, and M. A. Gerosa, "Applying large language models api to issue classification problem," *arXiv preprint arXiv:2401.04637*, 2024.

[5] J. Heo and S. Lee, "A study on applying large language models to issue classification," in *2025 IEEE/ACM 33rd International Conference on Program Comprehension (ICPC)*, pp. 1–11, 2025.

[6] G. Colavito, F. Lanubile, N. Novielli, and L. Quaranta, "Large language models for issue report classification," 2024.

[7] G. Colavito, F. Lanubile, and N. Novielli, "Few-shot learning for issue report classification," in *2023 IEEE/ACM 2nd International Workshop on Natural Language-Based Software Engineering (NLBSE)*, pp. 16–19, 2023.

[8] M. L. Siddiq and J. C. Santos, "Bert-based github issue report classification," in *2022 IEEE/ACM 1st International Workshop on Natural Language-Based Software Engineering (NLBSE)*, pp. 33–36, IEEE, 2022.

[9] N. Pandey, A. Hudait, D. K. Sanyal, and A. Sen, "Automated classification of issue reports from a software issue tracker," in *Progress in Intelligent Computing Techniques: Theory, Practice, and Applications: Proceedings of ICACNI 2016, Volume 1*, pp. 423–430, Springer, 2018.

[10] S. Bharadwaj and T. Kadam, "Github issue classification using bert-style models," in *2022 IEEE/ACM 1st International Workshop on Natural Language-Based Software Engineering (NLBSE)*, pp. 40–43, IEEE, 2022.

[11] H. Fazayeli, S. M. Syed-Mohamad, and N. S. M. Akhir, "Towards auto-labelling issue reports for pull-based software development using text mining approach," *Procedia Computer Science*, vol. 161, pp. 585–592, 2019.

[12] P. S. Kochhar, F. Thung, and D. Lo, "Automatic fine-grained issue report reclassification," in *2014 19th International Conference on Engineering of Complex Computer Systems*, pp. 126–135, IEEE, 2014.

[13] S. J. Dommati, R. Agrawal, S. S. Kamath, *et al.*, "Bug classification: Feature extraction and comparison of event model using na\" ive bayes approach," *arXiv preprint arXiv:1304.1677*, 2013.

[14] K. Goseva-Popstojanova and J. Tyo, "Identification of security related bug reports via text mining using supervised and unsupervised classification," in *2018 IEEE International conference on software quality, reliability and security (QRS)*, pp. 344–355, IEEE, 2018.

[15] X. Xie, Y. Su, S. Chen, L. Chen, J. Xuan, and B. Xu, "Mula: A just-in-time multi-labeling system for issue reports," *IEEE Transactions on Reliability*, vol. 71, no. 1, pp. 250–263, 2021.

[16] Y. Zhu, M. Pan, Y. Pei, and T. Zhang, "A bug or a suggestion? an automatic way to label issues," *arXiv preprint arXiv:1909.00934*, 2019.

[17] L. Schulte, B. Ledel, and S. Herbold, "Studying the explanations for the automated prediction of bug and non-bug issues using lime and shap," *Empirical Software Engineering*, vol. 29, no. 4, p. 93, 2024.

[18] A. Trautsch and S. Herbold, "Predicting issue types with sebert," in *Proceedings of the 1st International Workshop on Natural Language-based Software Engineering*, pp. 37–39, 2022.

[19] J. Von der Mosel, A. Trautsch, and S. Herbold, "On the validity of pre-trained transformers for natural language processing in the software engineering domain," *IEEE Transactions on Software Engineering*, 2022.

[20] M. T. Ribeiro, S. Singh, and C. Guestrin, "" why should i trust you?" explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1135–1144, 2016.

[21] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," *Advances in neural information processing systems*, vol. 30, 2017.

[22] X. Lu and J. Ma, "Does faithfulness conflict with plausibility? an empirical study in explainable ai across nlp tasks," *arXiv preprint arXiv:2404.00140*, 2024.

[23] A. Shrikumar, P. Greenside, and A. Kundaje, "Learning important features through propagating activation differences," in *International conference on machine learning*, pp. 3145–3153, PMLR, 2017.

[24] J. DeYoung, S. Jain, N. F. Rajani, E. Lehman, C. Xiong, R. Socher, and B. C. Wallace, "Eraser: A benchmark to evaluate rationalized nlp models," *arXiv preprint arXiv:1911.03429*, 2019.