Explainable Machine Learning Techniques for Criticality Prediction of Software Change Requests

Amanda Q. R. dos Santos[1], **José R. da Silva**[1], **Renata F. Lins**[1], **Ricardo B. C. Prudêncio**[1]

[1]Centro de Informática, Universidade Federal de Pernambuco
Recife, Brasil

Abstract. *Text classification is an essential task in several real applications, commonly involving the use of Natural Language Processing (NLP) and Machine Learning (ML). Specifically, in the context of software testing, text classifiers have been applied to the analysis of Change Requests (CRs) documents, which are reports that report errors found in the software under test. Prioritizing critical CRs is crucial for more effectively maintaining the quality of developed products. This article explores the use of eXplainable AI (XAI) algorithms for the inspection of CR criticality classifiers, facilitating the interpretation and justification of decisions made by the ML model. Once a CR is classified as critical, explanations are generated to identify important characteristics that help understand why the error reported by the CR was considered critical. In a case study, we implemented the LIME technique to provide interpretable explanations of the predictions made by an XGBoost model, applied to a database extracted from a real industrial context of mobile device development. The results show that XAI can increase confidence in ML model decisions by highlighting the most significant terms in critical and non-critical CRs.*

1. Introduction

Text classification has been applied in different contexts of application both with Natural Language Processing (NLP) and Machine Learning (ML) techniques. In the current work, text classification is applied for analysing Change Request (CR) documents, which are produced during software testing activities. CRs are free-text reports created by software testers when an unexpected behavior is observed in the software under test [Zhang et al. 2015]. Once created, the triage task is performed to identify the most critical CRs, usually associated to more impacting bugs in an application. This is a very important task to maintain the quality of a software, however it is expensive and prone to errors if totally performed by humans. Based on this motivation, text classifiers have been adopted in the literature to predict the criticality of CRs based on its textual context [Lamkanfi et al. 2010], thus supporting a more efficient triage process.

Although text classifiers can be actually useful to speed up the process of identifying critical CRs, the final decision about prioritizing or not a CR is in many cases made by a software engineer or manager. So it is essential to build not only accurate ML models, but also interpretable ones, in such a way that the software engineer can better understand why a certain CR was predicted as critical. The area of eXplainable Artificial Intelligence (XAI) [Molnar 2020] is crucial to meet this need. XAI aims to provide insights into how AI models work, highlighting the factors that influence their decisions

and providing clear and interpretable explanations for the outcomes they produce. This not only increases confidence in the decisions made by the AI systems, but also allows for the detection and correction of errors and potential biases in the models.

In this paper, we explore the use of XAI techniques in the context of CR criticality classification. For this, we developed a case study in which a criticality classifier was built in an industrial environment of mobile development at a project between Motorola Mobility (a Lenovo Company) and CIn-UFPE. Initially a corpus of 1441 CRs was collected, where each CR has been already associated to a criticality label. This labeled corpus is a result of the triage process periodically performed by the company. Based on the collected corpus, a standard process was adopted to build the criticality classifier, where NLP techniques were adopted to produce a vector representation of each CR and a ML algorithm was adopted to learn the predictive model. The built model was evaluated, obtained an area under the ROC curve around 0.7, which represents a positive result considering the application purposes.

For XAI, we adopted the LIME (Local Interpretable Model-agnostic Explanations) technique [Ribeiro et al. 2016] to identify the most important terms in each CR that explain the ML prediction. Additionally, we aggregated the explanations for sets of CRs in such a way to provide a more global explanation for the ML model. This global explanation was visualized as a word cloud, where it was possible to identify important contexts where CRs have been considered critical, such as errors found in specific and newer devices under test or related to priority functionalities (e.g., tests of 5G functionality).

The remaining os this paper is organized as follows. Section 2 brings a background and related work, followed by Section 3 where we described the developed case study. Section 4 in turn presents the performed experiments. Finally, Section 5 brings some conclusions and future works.

2. Theoretical Framework

In this section, we present a theoretical framework for understanding the developed work. Initially, we provide a brief discussion on the topic of XAI, followed by XAI for text classifiers. Finally, we present a discussion on the classification of CRs, which is our application scope.

2.1. Explainable Artificial Intelligence

Explainable Artificial Intelligence (XAI) refers to a range of methods and techniques that explain the decisions, processes, and functioning of an AI system, and specifically a machine learning (ML) system, in a manner comprehensible to humans. XAI techniques aim to provide insights into how AI models work, highlighting the factors influencing their decisions and offering clear and interpretable explanations for the results produced. This not only increases confidence in the decisions made by the model but also allows for the detection and correction of potential biases or errors in them. In this regard, understanding the motivations behind predictions can help learn more about the problem, the data, and the reasons why a model might fail [Molnar 2020].

XAI techniques can be categorized in various ways. *Global* techniques provide more general explanations about the behavior of a system. An example is the Feature Im-

portance technique [Fisher et al. 2019], used to quantify the importance of each predictor variable used by supervised ML models. *Local* techniques, on the other hand, generate explanations for specific instances of a problem. For example, the LIME technique, or *Local Interpretable Model-agnostic Explanations*, is an explanation technique that clarifies the predictions of any classifier by learning an interpretable model locally around a given instance [Ribeiro et al. 2016]. LIME can be applied to generate explanations for the prediction returned for each instance of interest. In this case, it is also possible to identify important predictor variables, but locally, i.e., specifically used by the model for predictions in the region around the instance of interest.

2.2. XAI for Text Classification

Text classification typically involves NLP and ML techniques. NLP techniques are generally used to generate a vector representation of each textual document, involving, for example, tokenization operators, removal of irrelevant words (stopwords), stemming, and the use of word embeddings. Traditional machine learning algorithms can be evaluated on various metrics, such as accuracy, F1-score, recall, and precision. These metrics measure the model's performance; however, none of them can be used to interpret the results in a language comprehensible to humans [Zahoor et al. 2024].

Several explainable ML methods have been developed to meet the need for providing interpretability of ML models. In the context of NLP problems, there are tools such as SHapley Additive exPlanations (SHAP), a unified interpretability (both *local* and *global*) framework with a rigorous theoretical foundation on the game-theoretic concept of Shapley values [Lundberg and Lee 2017]. Furthermore, the Bayesian framework MARTA (MApping human Rationales To Attention) promotes XAI by unifying a human specialist knowledge with an attention-based model parameters. Its parameters and person reliability are updated in an iterative manner, allowing their learning processes to benefit from each other until agreements on both the label and rationales are reached [Arous et al. 2021].

Among all options, LIME was selected for the application in question. It approximates the model locally by employing a linearly weighted combination of input features to explain individual predictions, whilst using less computational resources than the others.

2.3. CR Classification

CRs are, in general terms, tickets opened by operators or test engineers after identifying undesirable behaviors in a device, whether hardware or software. They include information about the occurrence and description of the error, instructions on how to reproduce it, the date of occurrence, the products and components involved, among other relevant details.

In this context, each created CR generates a new task for the development team, which must work to resolve the problem in the best possible way. However, some challenges arise, such as determining which tickets should be treated with higher urgency and which can wait. Usually, this prioritization task is performed by a development leader, but over time, this responsibility can become exhausting and harm the team's productivity [Gomes et al. 2019]. Therefore, there is a natural curiosity to employ machine learning techniques to try to automate the process of classifying the criticality level of each CR.

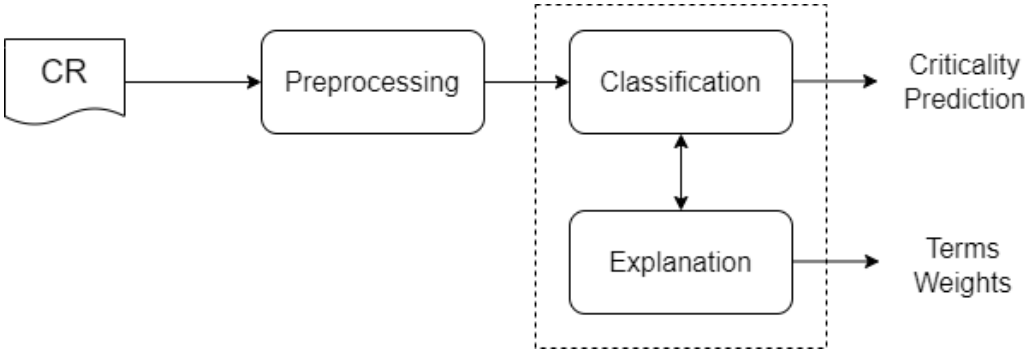

Figure 1. CR Classification and Explanation Pipeline

3. XAI for Criticality Classifiers of CRs

As seen in the previous chapter, text classifiers can be useful for analyzing CRs in various contexts. Recently, ML algorithms were used for classifying the criticality of CRs in an industrial context of mobile device development[1]. The experiments showed positive results in terms of predictive performance metrics. Once a good criticality classifier is built, it can be integrated into the CR triage process, identifying the errors that should be prioritized for resolution by developers.

In the present work, we explore a line of investigation that involves the use of explainability techniques for criticality classifiers. The main motivation is to provide users (e.g., test engineers and developers) with a deeper understanding of the reasons that make a CR critical or not. More specifically, the use of XAI in this context aims to identify characteristics present in the CR texts that would make them more critical according to the classification model. Thus, the explanations generated by XAI techniques would help clarify and justify the decisions made in the process of triaging critical CRs.

Figure 1 presents the general architecture of the developed solution. Given a CR as input, its textual content is preprocessed in such a way to produce a vector representation. Then, the Classification module receives the CR's representation and returns a criticality prediction. This module brings a ML model previously learned by using a training set with CRs labeled as either critical or non-critical. Finally the Explanation module is integrated to the classifier in order to provide explanations for the returned prediction. In our work, we adopt the LIME technique to return the importance weight associated to each CR's term. Eventually, such explanations could be aggregated for a set of CRs given as input in such a way to provide a global assessment of the importance of each term.

In our work, we performed a case study in which a classifier was built by using a set of 1441 CRs collected in an industrial environment. Next section will present the dataset adopted, followed by the implementation details of each module in the proposed architecture.

3.1. Dataset

The classification and explainability methods were applied to a private dataset extracted directly from JIRA, which contains information on 1,441 past change requests that were created between June 8, 2020 and May 4, 2022, already labeled as critical or non-critical,

[1]Previous work omitted due to the blind review policy.

Table 1. XGBoost model parameters.

Parameter	Value
scale_pos_weight	1
learning_rate	0.0008
colsample_bytree	0.3
subsample	0.85
objective	binary:logistic
n_estimators	700
reg_alpha	0.3
max_depth	3
gamma	5

with 437 of these change requests considered critical and 1004 non-critical. This information includes the names of the affected device and component, creation date, priority, and type of issue. Among this information, the most important for this research and effectively used for the application in question are: (1) the summary, which provides a brief explanation of the error reported in the CR; and (2) the class, which is pre-filled during the triage process and indicates whether the CR is critical or not.

3.2. Text Pre-processing

Each CR summary in the dataset is processed using text operators to obtain a vector representation for classification. In this work, the following text operators were applied:

- Conversion of all uppercase letters to lowercase;
- Removal of special characters, including non-ASCII characters, line breaks, and excessive spaces;
- Tokenization, which splits the text into tokens, consisting of all terms separated by spaces and other types of tabulation;
- Removal of English stopwords, such as *I*, *we* and *the*, except for some words like *no*, *not* and *don't*, as well as the removal of specific words from the application context that do not influence the determination of the CR's criticality;
- Lemmatization and stemming to reduce inflections and variations of words to their common root.

The above operators ensure that each text is clean and consistent, resulting in better performance in the classification task a resulting dataset of 231 feature columns and 1 class column.

3.3. Classification Module

In this module, the XGBoost algorithm was chosen, which has proven to be competitive in various application contexts, requiring minimal computational resources for training, and it also has extensive support for explainability tools within the community. Table 1 presents the parameter values used for the algorithm in this case study. For any parameters not mentioned, the default values from the XGBoost library [XGBoost Developers 2022] were used.

Since the number of critical CRs is usually smaller, class balancing techniques were used to optimize the performance of the classification model:

- ADASYN [He et al. 2008]: Similar to SMOTE, but with a greater focus on more difficult and less represented minority class data, creating adaptive synthetic instances. It will generate new data until the dataset is balanced.
- SMOTE [Chawla et al. 2002]: Generates new data belonging to the minority class based on its nearest neighbors. It will generate new data until the dataset is balanced.
- RandomOverSampler [Mohammed 2020]: Replicates data from the minority class until the dataset is balanced.

3.4. Explanation Module

To provide explainability for the trained model, a module was created consisting of two steps: generating explanations using LIME and aggregating local importance for visualizing the most significant terms.

For each individual test CR, LIME can be applied to determine the importance of each term in the criticality classification. For each term t_i and test instance d_j, LIME returns a numerical weight w_{ij}, which indicates the importance of that term for the classification of the instance. The local surrogate model used in LIME is a logistic regression model, and the importance of the terms is represented by the model's weights. The higher the absolute value of the weight, the more significant the analyzed term is. A positive weight indicates that the term is important for classifying the CR as critical, while a negative weight indicates that the term is important for classifying the CR as non-critical.

In this module, we aggregate the weights returned by LIME to obtain a global importance on a test dataset. Unlike global importance methods (e.g., permutation feature importance), this strategy does not require labeled test data. Given a test dataset T, the global weight of term t_i is defined as:

$$\mathbf{w}_i = \sum_{d_j \in T} w_{ij} \tag{1}$$

The value of $\mathbf{w}_i$ is influenced by the number of times the term appears among the explanations of the test examples, as well as the individual weights associated. The global weight indicates the significance of the term in the criticality classification task for CRs. If the test dataset is labeled, the global weight can be decomposed by CR class:

$$\mathbf{w}_i^{(1)} = \sum_{d_j \in T^{(1)}} w_{ij} \tag{2}$$

$$\mathbf{w}_i^{(0)} = \sum_{d_j \in T^{(0)}} w_{ij} \tag{3}$$

where $T^{(1)}$ is the subset of test instances belonging to the critical class and $T^{(0)}$ is the subset of test instances belonging to the non-critical class.

4. Experiments

Initially, in the experiments, we evaluated the predictive performance of the XGBoost model in the classification task. To do this, we split the dataset into a training set, with

80% of the examples, and a test set with the remaining examples. Each class balancing method was also evaluated in this experiment 2. It is important to note that the balancing technique is applied only to the training data to avoid any contamination or positive bias in the test set. For each trained model, we used the area under the ROC curve (AUC) as the evaluation metric. This metric was chosen in our context because it evaluates the ranking of examples ordered by estimated class probabilities, i.e., it assesses the quality of a ranking of CRs ordered by the probability of being critical.

Table 2. XGBoost model parameters.

Method	AUC Value
None	0.69
RandomOverSampler	0.73
SMOTE	0.59
ADASYN	0.58

Table 2 presents the obtained AUCs, which ranged from 0.58 to 0.73, depending on the balancing method. Figure 2 presents the ROC curve obtained from the best result achieved (AUC = 0.73). Notice that in the initial segment of the ROC curve, we have, for example, a true positive rate (TPR) of 0.5 (i.e., 50% of critical CRs correctly retrieved) obtained at a false positive rate (FPR) of around 0.2. This scenario would be plausible only in cases where the time available for CR triage is very short, so the goal is to minimize the number of FPs while maintaining a reasonable TPR. At the other extreme of the ROC curve, a TPR of 0.9 is obtained when the FPR is around 0.8. Although an FPR of 0.8 is high in itself, it represents a 20% reduction in the number of non-critical CRs analyzed during triage.

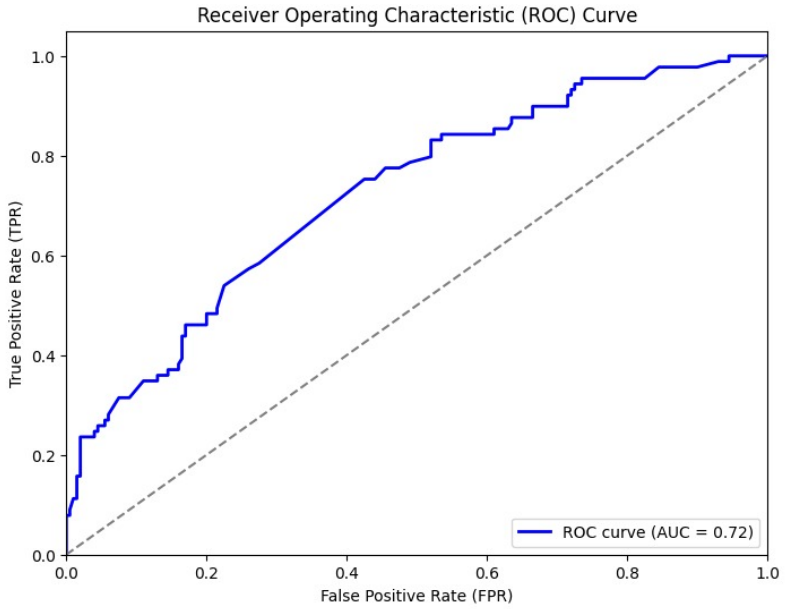

Figure 2. Area Under the ROC Curve (AUC).

Finally, based on the results obtained from the previous step, a simple word cloud visualization can be used to identify the most important terms for critical and non-critical

Figure 3. Most relevant words in critical CRs.

test CRs, respectively (see Figure 3). The size of each term t_i in the word cloud is proportional to the global weights, $\mathbf{w}_i^{(0)}$ and $\mathbf{w}_i^{(1)}$, defined by class.

In Figure 3, for example, we can see a significant relevance of the term *Device X*[2], which is one of the devices tested by the testing team, and *plus*, a term often used as a suffix to indicate an enhanced or feature-rich version of a device compared to the regular version. This indicates that errors found in this device are prioritized for correction. Another important term is "5G", which signifies a new functionality that is a priority for the company.

5. Conclusion

In this paper we presented the use of text classifiers to predict the criticality of CRs. Specifically, we adopt the use of XAI techniques to explain the predictions returned by the ML model. From the experiments conducted and the results obtained, it is evident that there are patterns in the content of a CR that can help classify them as critical or non-critical, in the form of words present in their summaries. In this context, certain terms will receive higher weights for the model's classification, either due to a device that frequently fails, such as *Sofia*, or due to mentions of critical systems for the applications, such as 5G.

Therefore, the use of Explainable AI combined with machine learning techniques presents itself as a promising solution for automating the analysis and prioritization of Change Requests, ensuring the continuous and agile improvement of product and service quality at the partner company.

Future research can explore the application of other XAI techniques and identify additional patterns that influence the classification of CRs, using large language models and other machine learning algorithms.

References

Arous, I., Dolamic, L., Yang, J., Bhardwaj, A., Cuccu, G., and Cudré-Mauroux, P. (2021). Marta: Leveraging human rationales for explainable text classification. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(7):5868–5876.

[2]Device name omitted due to compliance.

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

Fisher, A., Rudin, C., and Dominici, F. (2019). All models are wrong, but many are useful: Learning a variable's importance by studying an entire class of prediction models simultaneously. *Journal of Machine Learning Research*, 20(177):1–81.

Gomes, L. A. F., da Silva Torres, R., and Côrtes, M. L. (2019). Bug report severity level prediction in open source software: A survey and research opportunities. *Information and Software Technology*, 115:58–78.

He, H., Bai, Y., Garcia, E. A., and Li, S. (2008). Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In *2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence)*, pages 1322–1328. Ieee.

Lamkanfi, A., Demeyer, S., Giger, E., and Goethals, B. (2010). Predicting the severity of a reported bug. In *2010 7th IEEE working conference on mining software repositories (MSR 2010)*, pages 1–10. IEEE.

Lundberg, S. M. and Lee, S.-I. (2017). A unified approach to interpreting model predictions. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Mohammed, Roweida e Rawashdeh, J. e. A. M. (2020). Aprendizado de máquina com técnicas de sobreamostragem e subamostragem: estudo geral e resultados experimentais. In *2020 11ª Conferência Internacional sobre Sistemas de Informação e Comunicação (ICICS)*, pages 243–248.

Molnar, C. (2020). *Interpretable machine learning*. Lulu. com.

Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). "why should i trust you?": Explaining the predictions of any classifier.

XGBoost Developers (2022). Xgboost parameters. https://xgboost.readthedocs.io/en/stable/parameter.html, Acessado em 2024-02-06.

Zahoor, K., Bawany, N., and Qamar, T. (2024). Evaluating text classification with explainable artificial intelligence. *IAES International Journal of Artificial Intelligence (IJ-AI)*, 13:278–286.

Zhang, J., Wang, X., Hao, D., Xie, B., Zhang, L., and Mei, H. (2015). A survey on bug-report analysis. *Sci. China Inf. Sci.*, 58(2):1–24.