http://doi.org/10.35784/iapgos.3396 *received: 30.12.2022 | revised: 12.03.2023 | accepted: 15.03.2023 | available online: 31.03.2023*

APPLICATION OF EXPLAINABLE ARTIFICIAL INTELLIGENCE IN SOFTWARE BUG CLASSIFICATION

Lukasz Chmielowski[1,2], Michal Kucharzak[1,2], Robert Burduk[2]
[1]Nokia Solutions and Networks sp. z o.o., Warsaw, Poland, [2]Wroclaw University of Science and Technology, Faculty of Information and Communication Technology, Wroclaw, Poland

Abstract. *Fault management is an expensive process and analyzing data manually requires a lot of resources. Modern software bug tracking systems may be armed with automated bug report assignment functionality that facilitates bug classification or bug assignment to proper development group. For supporting decision systems, it would be beneficial to introduce information related to explainability. The purpose of this work is to evaluate the use of explainable artificial intelligence (XAI) in processes related to software development and bug classification based on bug reports created by either software testers or software users. The research was conducted on two different datasets. The first one is related to classification of security vs non-security bug reports. It comes from a telecommunication company which develops software and hardware solutions for mobile operators. The second dataset contains a list of software bugs taken from an opensource project. In this dataset the task is to classify issues with one of following labels crash, memory, performance, and security. Studies on XAI-related algorithms show that there are no major differences in the results of the algorithms used when comparing them with others. Therefore, not only the users can obtain results with possible explanations or experts can verify model or its part before introducing into production, but also it does not provide degradation of accuracy. Studies showed that it could be put into practice, but it has not been done so far.*

Keywords: software bug assignment, software bug triaging, explainable artificial intelligence, text analysis, vulnerability

ZASTOSOWANIE WYJAŚNIALNEJ SZTUCZNEJ INTELIGENCJI W KLASYFIKACJI USTEREK OPROGRAMOWANIA

Streszczenie. *Zarządzanie usterkami jest kosztownym procesem, a ręczna analiza danych wymaga znacznych zasobów. Nowoczesne systemy zarządzania usterkami w oprogramowaniu mogą być wyposażone w funkcję automatycznego przypisywania usterek, która ułatwia klasyfikację usterek lub przypisywanie usterek do właściwej grupy programistów. Dla wsparcia systemów decyzyjnych korzystne byłoby wprowadzenie informacji związanych z wytłumaczalnością. Celem tej pracy jest ocena możliwości wykorzystania wyjaśnialnej sztucznej inteligencji (XAI) w procesach związanych z tworzeniem oprogramowania i klasyfikacją usterek na podstawie raportów o usterkach tworzonych przez testerów oprogramowania lub użytkowników oprogramowania. Badania przeprowadzono na dwóch różnych zbiorach danych. Pierwszy z nich związany jest z klasyfikacją raportów o usterkach związanych z bezpieczeństwem i niezwiązanych z bezpieczeństwem. Dane te pochodzą od firmy telekomunikacyjnej, która opracowuje rozwiązania programowe i sprzętowe dla operatorów komórkowych. Drugi zestaw danych zawiera listę usterek oprogramowania pobranych z projektu opensource. W tym zestawie danych zadanie polega na sklasyfikowaniu problemów za pomocą jednej z następujących etykiet: awaria, pamięć, wydajność i bezpieczeństwo. Badania przeprowadzone przy użyciu algorytmów związanych z XAI pokazują, że nie ma większych różnic w wynikach algorytmów stosowanych przy porównywaniu ich z innymi. Dzięki temu nie tylko użytkownicy mogą uzyskać wyniki z ewentualnymi wyjaśnieniami lub eksperci mogą zweryfikować model lub jego część przed wprowadzeniem do produkcji, ale także nie zapewnia to degradacji dokładności. Badania wykazały, że można to zastosować w praktyce, ale do tej pory tego nie zrobiono.*

Słowa kluczowe: przypisywanie usterek oprogramowania, klasyfikacja usterek oprogramowania, wyjaśnialna sztuczna inteligencja, analiza tekstu, podatności

Introduction

For large scale software development many tools related to the environment are usually used including among others code repositories, bug tracking systems or decision support systems. Part of them might use machine learning predictions. They are supporting or providing different decisions like assigning priority, severity, group to investigate or solve problem, or label issue as security related or not. An example of black-box model application for identifying security bugs is described in publication [9]. In contrast to black-box solutions a proposition of application of one based on expert rules is shown in paper [2]. Bug mining tool to identify and analyze security bugs using naive bayes and tf-idf was shown in International Conference on Reliability Optimization and Information Technology [4]. Both methods used allow solution to be explainable, but this circumstance was not used. The main aim was to analyse possibilities of application of the explainable artificial intelligence (XAI) in specific cases related to software development.

There are two major taxonomies related to explainability of Machine Learning (ML) models. The first, related to distinction between transparency (including models that are transparent by design), including post-hoc explainability. The second taxonomy, which concerns XAI methods tailored to explain deep learning models. In this context, XAI uses classification criteria based on ML techniques, e.g., representation vectors, layerwise attention [3]. As the first taxonomy is more general and extensive, it is used as a baseline definition of XAI in this document. General review on XAI and its various applications can be found in material [16].

According to the best knowledge of authors there is no application of explainable artificial intelligence techniques in neither solving the problem of assigning security labels nor group responsible for investigating or solving software bug. Nevertheless, there are articles related to possible applications of XAI techniques and their benefits into a system that suggests patches into source code. According to authors of publication [14] in cases where proposed patches are provided without explanations they are usually ignored. In that paper was a statement that those kinds of systems which support developers in way that it can be explained to them is a future of supporting tools in software development.

Another application of explainable artificial intelligence in software development was found in paper [11]. There is a description of works related to predict whether the software commit is risky. To explain it uses predefined features extracted from commits like among others number of modified lines, files, subsystems, and information if change was related to fixing defect. In article [1] are shown results of application of model agnostic explanation methods like LIME and iBreak on bug prediction models. Paper [10] presents assigning the bug severity level. Even if it is not strictly related to XAI methods, at one of steps it is uses algorithm based on dictionary of critical terms related to appropriate severity level. That information might be useful to support the creation of expert systems to support or provide such decisions when there is lack of trust in black-box models.

Explainable artificial intelligence systems might be applied in cases where there is special need for trust in model predictions especially in safety critical applications [6]. Part of those white-box models make the option to generate rules which might be verified by experts with domain knowledge possible. Those kinds of solutions might be useful especially in systems with high responsibility. One of methods of extracting rules which might be verified by experts is to use univariate tree classifier. Then the tree structure might be inspected. Another use case is to provide expert support by providing decision of model with explainable rule extracted from tree. Example of publication with use of such trees is paper [13]. For extracting rules with a decision tree from black-box model as example may be an article [5]. The research questions which are being answered in the work are presented below:

- What is accuracy when comprising standard and easily explainable algorithms?
- What benefits might that gives?
- Does information provided by explainable models seem to be consistent?

1. Methods

The data used for research comes from two different sources:

1. Internal company data:
   The purpose of internal data is to distinguish between security and non-security issues. Due to trade secrets no details about the quantity of samples could be provided, but information about distribution of data is shown in table 1. Another example of article using data comes also from NOKIA is publication [8].

Table 1. Diagram of tree build on dataset from Mozilla to recognize types of issues

Type of issue	Percent of reports
Security related issues	4.1 %
Non-security related issues	95.9 %

2. Mozilla Defect Dataset:
   Data from Mozilla is widely available. It contains software bugs labeled as performance problem, security related issue or crash, memory. Details with quantity of samples of dataset extracted for this publication are shown in table 2. Samples with multiple labels were removed. Generally, publicly available bug reports from Mozilla projects are accessible, among others, in repository [12] or can be gathered with script [7].

Table 2. Distribution of type od issue (Mozilla Defect Dataset)

Type of issue	Percent of reports
Crash related issue	66.3 %
Memory related issue	11.4 %
Security related issue	11.2 %
Performance related issue	11.1 %

For chosen selected classification not found any publication which has data to comparison. The selection of those specific datasets is justified by the fact that relatively no such deep domain knowledge is required to interpret those cases. Research has been carried out in both cases according to the same experimental protocol. Firstly, on raw text data extracted from title of cases was performed preprocessing contains among others, removing special characters, stopwords then applying lemmatization. In the next step vectorization was applied with usage of tf-idf with limitation of max features parameter to 1000. Features which were taken into consideration by tf-idf are both unigrams and bigrams. On data prepared that way calculations were performed with usage of different algorithms. Results of selected standard algorithms used for XAI applications were compared against rest which were introduced. The method to explain results was univariate tree to extract the rules. Moreover, most important features according to different models were extracted to be compared in subjective way. That extraction may potentially be used in context of creation expert rules.

2. Results and discussion

Comparison of results with usage of both types of algorithms which can be used straightforward to as explainable and not are shown in tables 3–8. Headings used in tables are:

- kNN – k – Nearest Neighbors;
- LR – Logistic Regression;
- NB – Naive Bayes;
- SVC – Support Vector Classifier;
- XGBoost – eXtreme Gradient Boosting;
- tree x-y-z – Decision Tree Classifier where:
  - x – minimum number of samples required to split an internal node leaf,
  - y – minimum number of samples required to split an internal node,
  - z – maximum depth of tree.

Table 3. Comparison most important features related to label issue as security related or not (internal company data) in condition of selected algorithm

tree 5-5-15	LR	SVC	XGBoost
vulnerability	vulnerability	vulnerability	vulnerability
sensitive	security	sensitive	sensitive
security	svm	security	security
svm	sensitive	svm	svm
password	sec	scan	password

As is shown in table 3 most important features for chosen classifiers related to task to distinguish if case is security related or not are: vulnerability, svm, sensitive, security. For use case related to label issue as performance, security, crash, or memory related problem following terms are most important: crash, leak, memory (table 4). It is noticeable in both of cases that at least some of the same features are common for most important classifiers. This is also confirmed in figure 1 and figure 2. Diagrams (figures 1–3) present decision making process, how the classification is performed with the use of decision trees. Each of them shows a section of the decision tree related to one of the discussed problems. Analyzing the content of diagram in figure 3, the root node is shown at the top. The first text line of that node indicates that the decision depends on frequency of occurrence of keyword *vulnerability*. It is shown that in that case, if value of parameter related to *vulnerability* is above threshold, the condition for node is False. Therefore, following the arrow (branch) marked False, the next node is selected. It has the majority class Yes, what means it is related to security as it was expected. As that one node is not a leaf node, algorithm follows the next conditions. Color of node which is used for presentation depends on the purity of the node. In this example security related issues are in blue and non-security related ones are in orange. There is also presented a measure of impurity which is in that case Gini.

Table 4. Comparison most important features related to type of issue (Mozilla Defect Dataset) in condition of selected algorithm

tree 5-5-15	LR	SVC	XGBoost
crash	crash	crash	crash
regression	application	leak	leak
content	intermittent	memory	regression
memory	leak	usage	addresssanitizer
slow	moz_crash	lazily	build

Results between decision tree as one which is interpretable by design (transparent model) and support vector classifier which requires external XAI techniques to be explained (post-hoc explainability) were used for explainability comparison.

With significance threshold at $\alpha = 0.05$, performing paired *t* test 5x2cv procedure returned $p - value = 0.19$. As $p - value > \alpha$, the null hypothesis cannot be rejected, and it may be concluded that the performance of the two algorithms is not significantly different. That is expected as we gain explainability, without loss of quality of results. More details about used test procedure is in [15].

Table 5. Comparison of results related to label issue as security related or not (internal company data)

	kNN		LR		NB		SVC		XGBoost	
class	prec	recall	prec	recall	prec	recall	prec	recall	prec	recall
Security related	0.96	0.84	0.98	0.79	0.32	0.90	0.97	0.85	0.99	0.76
Non-security related	0.99	1.00	0.99	1.00	1.00	0.92	0.99	1.00	1.00	0.92

Table 6. Comparison of results related to label issue as security related or not (internal company data)

	tree 5-5-10		tree 5-5-15		tree 10-10-15		tree 3-3-15		tree 5-5-5	
class	prec	recall	prec	recall	prec	recall	prec	recall	prec	recall
Security related	0.93	0.86	0.93	0.86	0.92	0.82	0.95	0.84	0.96	0.79
Non-security related	0.99	1.00	0.99	1.00	0.99	1.00	0.99	1.00	0.99	0.99

Table 7. Comparison of results related to type of issue (Mozilla Defect Dataset)

	kNN		LR		NB		SVC		XGBoost	
class	prec	recall	prec	recall	prec	recall	prec	recall	prec	recall
Crash	0.88	0.94	0.97	0.97	0.97	0.47	0.98	0.96	0.98	0.95
Memory	0.72	0.38	0.83	0.46	0.12	0.65	0.78	0.59	0.69	0.33
Performance	0.72	0.65	0.83	0.88	0.76	0.68	0.83	0.88	0.97	0.49
Security	0.52	0.47	0.70	0.74	0.25	0.69	0.69	0.74	0.38	0.85

Table 8. Comparison of results related to type of issue (Mozilla Defect Dataset)

	tree 5-5-10		tree 5-5-15		tree 10-10-15		tree 3-3-15		tree 5-5-5	
class	prec	recall	prec	recall	prec	recall	prec	recall	prec	recall
Crash	0.98	0.95	0.98	0.95	0.98	0.95	0.98	0.95	0.98	0.92
Memory	0.77	0.33	0.77	0.33	0.69	0.30	0.80	0.31	0.78	0.17
Performance	0.99	0.59	0.99	0.59	0.99	0.59	0.99	0.59	0.98	0.43
Security	0.41	0.87	0.99	0.87	0.41	0.87	0.41	0.90	0.33	0.87

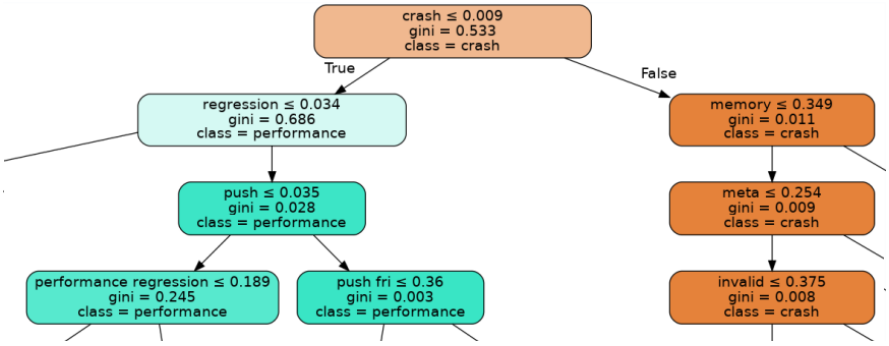

Fig. 1. Diagram of tree build on dataset from Mozilla to recognize types of issues

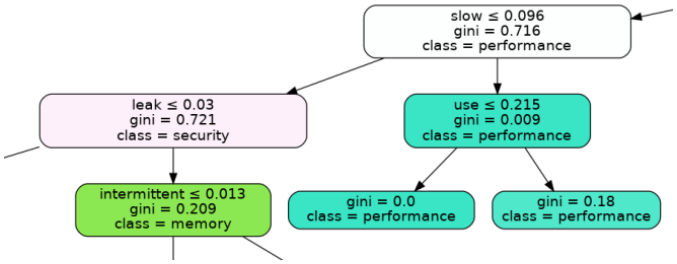

Fig. 2. Diagram of tree build on dataset from Mozilla to recognize types of issues

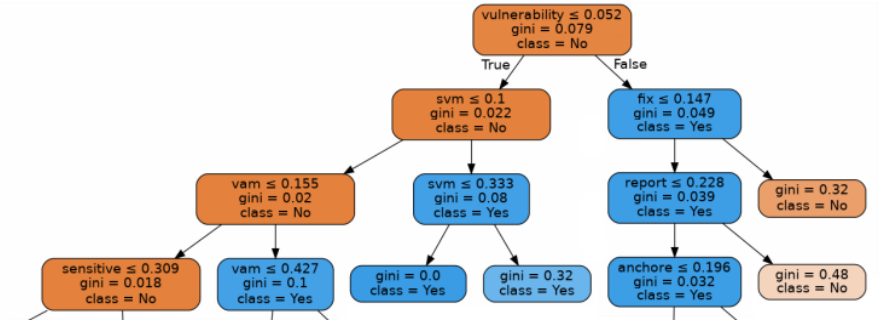

Fig. 3. Diagram of tree built on internal company dataset to recognize security related issues

3. Conclusion

The paper discusses potential application of the explainable artificial intelligence in software bug report classification. In the article the authors discuss the possibility of using XAI methods in the context of assigning a department, group, or any label for software bug reports created by the user or testers. According to authors there is currently no application of such solution, however there are papers which consider different, but sometimes confused with the mentioned problem. That similar, but significantly different topic is software bug prediction, which aims to indicate whether introducing software change will lead to a defect. The presented results show experimental research with the use of simulations of predictions of type of software bug or classify the issue as security related or not. One of the steps in the research was to apply explainable artificial intelligence methods and compare results between standard black-box methods and XAI ones. The result of comparison on Mozilla data shows that it can be useful. When applying XAI methods on dataset with company internal data it can be clearly noticed that rules generated seem to be legit and might be potentially used for explaining decisions or suggestions. For both cases there have been gathered most important features according to the trained models. In the presented diagrams (figures 1–3) the way how chosen built models make the decisions are shown. For one algorithm shown that has been applied, the decision-making process is shown. For each step (node) a decision condition is presented, what is the main class of samples that meets the specified conditions of the current node. To sum up this paper clearly shows that there is a possibility to apply explainable artificial intelligence methods in the context of problems related to bug assignment and the results are reasonable.

Author contributions

Author contributions statement L.C. independently conceived the experiments; L.C., M.K., R.B. analyzed results; L.C. wrote original draft; M.K., R.B. provided editorial suggestions; L.C., M.K. conducted editing of work; L.C., M.K., R.B. attempted to disprove the novelty. All authors reviewed the manuscript.

Acknowledgements

This work has been carried out in cooperation between NOKIA and Wroclaw University of Science and Technology in context of a Ph.D. grant under the fourth edition of the Implementation Doctorate Programme.

Conflicts of interests

The authors declare no conflicts of interests.

References

[1] Aleithan R.: Explainable Just-In-Time Bug Prediction: Are We There Yet? 43rd International Conference on Software Engineering: Companion Proceedings (ICSE-Companion), 2021, 129–131 [http://doi.org/10.1109/ICSE-Companion52605.2021.00056].
[2] Anjali, Mohan D., Sardana N.: Visheshagya: Time based expertise model for bug report assignment. Ninth International Conference on Contemporary Computing (IC3), 2016, 1–6 [http://doi.org/10.1109/IC3.2016.7880218].
[3] Barredo Arrieta A. et al.: Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. Information Fusion 58, 2020, 82–115 [http://doi.org/10.1016/j.inffus.2019.12.012].
[4] Behl D., Handa S., Arora A.: A bug Mining tool to identify and analyze security bugs using Naive Bayes and TF-IDF. International Conference on Reliability Optimization and Information Technology (ICROIT), 2014, 294–299 [http://doi.org/10.1109/ICROIT.2014.6798341].
[5] Carlevaro A., Maurizio M.: A New SVDD Approach to Reliable and Explainable AI. IEEE Intelligent Systems 37.2, 2022, 55–68 [http://doi.org/10.1109/ACCESS.2022.3180026].
[6] Carlevaro A. et al.: Counterfactual Building and Evaluation via eXplainable Support Vector Data Description. IEEE Access 10, 2022 [http://doi.org/10.1109/MIS.2021.3123669].
[7] Castelluccio M. et al.: bugbug. Available online: https://github.com/mozilla/bugbug (accessed on 02.11.2022).
[8] Chmielowski L., Kucharzak M.: Impact of Software Bug Report Preprocessing and Vectorization on Bug Assignment Accuracy. Progress in Image Processing, Pattern Recognition and Communication Systems. Edited by Michal Choraś, et al.: Springer International Publishing, Cham 2022, 153–162 [http://doi.org/10.1007/978-3-030-81523-3_15].
[9] Choquette-Choo C. A. et al.: A Multi-label, Dual-Output Deep Neural Network for Automated Bug Triaging. 18th IEEE International Conference On Machine Learning And Applications (ICMLA), 2019, 937–944 [http://doi.org/10.1109/ICMLA.2019.00161].
[10] Gujral S., et al.: Classifying bug severity using dictionary based approach. International Conference on Futuristic Trends on Computational Analysis and Knowledge Management (ABLAZE), 2015, 599–602 [http://doi.org/10.1109/ABLAZE.2015.7154933].
[11] Khanan C. et al.: JITBot: An Explainable Just-In-Time Defect Prediction Bot. 35th IEEE/ACM International Conference on Automated Software Engineering (ASE), 2020, 1336–1339.
[12] Lamkanfi A., Pérez J., Demeyer S.: The Eclipse and Mozilla defect tracking dataset: A genuine dataset for mining bug information. 10th Working Conference on Mining Software Repositories (MSR), 2013, 203–206 [http://doi.org/10.1109/MSR.2013.6624028].
[13] Matzka S.: Explainable Artificial Intelligence for Predictive Maintenance Applications. Third International Conference on Artificial Intelligence for Industries (AI4I), 2020, 69–74 [https://doi.org/10.1109/AI4I49448.2020.00023].
[14] Monperrus M.: Explainable Software Bot Contributions: Case Study of Automated Bug Fixes. IEEE/ACM 1st International Workshop on Bots in Software Engineering (BotSE), 2019, 12–15 [http://doi.org/10.1109/BotSE.2019.00010].
[15] Raschka S.: 5x2cv paired ttest. Available online: https://rasbt.github.io/mlxtend/user_guide/evaluate/paired_ttest_5x2cv (accessed on 04.01.2021).
[16] Vilone G. Longo L.: Explainable Artificial Intelligence: a Systematic Review. 2020 [http://doi.org/10.48550/arXiv.2006.00093].

M.Sc. Eng. Łukasz Chmielowski
e-mail: lukasz.chmielowski@nokia.com

Łukasz Chmielowski received a M.Sc. degree with distinction in Computer Science with specialization in Intelligent Information Systems. He is currently working towards Ph.D. in Information and Communication Technology at Wroclaw University of Science and Technology (Poland). He is with the Nokia Solutions and Networks sp. z o.o. (Poland) for five years. He is working with machine learning techniques related to natural language processing and software bug assignment.

http://orcid.org/0000-0001-6970-8144

Ph.D. Eng. Michał Kucharzak
e-mail: michal.kucharzak@pwr.edu.pl

Michał Kucharzak received his Ph.D. in computer science in area of network optimization. In recent years, he cooperated with numerous R&D centers and has been a member of reviewer committees for many international journals, program, and technical committees for various conferences as well. His current research interests are primarily in the areas of network modeling and network optimization with special regard to overlays, simulations, design of efficient algorithms and wireless system protocols, including software testing and quality assurance.

http://orcid.org/0000-0001-5068-5229

D.Sc. Eng. Robert Burduk
e-mail: robert.burduk@pwr.edu.pl

Robert Burduk is professor of Computer Science in the Department of Systems and Computer Networks, Faculty of Information and Communication Technology, Wroclaw University of Science and Technology, Poland. He received an Ph.D. and D.Sc. degrees in Computer Science in 2003 and 2014 respectively. His research interests cover among the others: machine learning, classifier selection algorithms and multiple classifier systems. He serves on program committees of numerous international conferences, published over 100 papers and edited 5 books.

http://orcid.org/0000-0002-3506-6611