Politechnika Wrocławska

FIELD OF SCIENCE:

ENGINEERING AND TECHNOLOGY

DISCIPLINE OF SCIENCE:

INFORMATION AND COMMUNICATION TECHNOLOGY

DOCTORAL DISSERTATION

APPLICATION OF MULTIMODAL NEURAL NETWORKS IN SOLVING PROBLEM OF LABELING BUG REPORTS

Łukasz Chmielowski, MSc, Eng.

Supervisor:
Prof. Robert Burduk, DSc, PhD, Eng.

Assistant supervisor:
Michał Kucharzak, PhD, Eng.

Keywords:

Bug assignment, Bug triaging, Bug report, Software bug, Text analysis

WROCŁAW 2024

ABSTRACT

In large scale software development organizations, there are often deployed complex processes related to the handling of software bug reports. Such documents about system malfunctions usually contain a title and a description of the discrepancy in the behavior of the solution in relation to the expected one in the form of natural language. In addition, system information is attached in raw format or processed by tools that analyze it. The main problem is assigning a problem report to the proper organizational unit and determining whether a given bug report is related to a security risk, memory issue, or performance issue. The data used in the experimental studies comes from a set of reports of bugs from telecommunications operators or were reported inside a company which develops telecommunications equipment including software for it. Dossier concerns base transceiver stations (BTS).

The research confirmed the thesis stated that *there exists a method for automated assignment of a software bug report to appropriate development group, responsible for resolving the bug, which outperforms well-known methods for bug report assignment.*

A literature review was conducted. One of the important conclusions was to define the research gap which indicates the drawbacks of the currently used validation techniques and to propose an alternative one devoid of the above-mentioned negative features. Experiments using protocols utilizing different ways of building training and test sets showed a significant difference in results. An analysis of the possibility of using explainable artificial intelligence (XAI) has been performed.

A number of studies were carried out, both related to preprocessing, vectorization of source data and at various levels of the company's structure. As part of the implementation of the doctorate, a novel method of assigning reports in the context of the organization's composition has been proposed. On the basis of research on the effectiveness of selected machine learning algorithms, a decision was made, and a pilot solution was implemented in the company. Due to the fact that it was possible to collect both the predictions of the machine learning model and the decisions of humans under certain conditions, it was decided to conduct a comparative analysis of the results.

The applicability of multimodal neural networks as well as other ways of solving the problem of assigning reports was investigated. The key aspect was to prepare an appropriate representation of the input data as well as to design the required multimodal neural network architecture. The conducted studies showed the superiority of this method compared to the reference methods used.

Attestation about implementation works at company is placed in Appendix C. It contains information about the ongoing work related to the introduction of a multimodal neural network. Its first versions are already implemented. Currently, work is underway related to a better selection of features and dealing with missing data. In addition, it provides information on other implementations related to software bug reports.

Keywords: *Bug assignment*, *Bug triaging*, *Bug report*, *Software bug*, *Text analysis*.

STRESZCZENIE

W dużych organizacjach zajmujących się rozwojem oprogramowania występują często złożone procesy związane z obsługą raportów o defektach powstałych podczas jego wytwarzania. Takie dokumenty o awariach systemu, zazwyczaj zawierają tytuł i opis rozbieżności w zachowaniu rozwiązania w stosunku do oczekiwanego w formie języka naturalnego. Dodatkowo są załączane informacje systemowe w formacie surowym lub przetworzonym przez narzędzia je analizujące. Głównym zagadnieniem jest przypisanie raportu do odpowiedniej komórki organizacyjnej oraz określenie czy dany raport jest powiązany z zagrożeniem dla bezpieczeństwa, nieprawidłościami z pamięcią lub wydajnością. Wykorzystane w badaniach eksperymentalnych raporty pochodzą od operatorów telekomunikacyjnych bądź zostały zgłoszone w firmie zajmującej się wytwarzaniem oprogramowania i sprzętu telekomunikacyjnego. Dotyczą one stacji bazowych (BTS).

Badania potwierdziły tezę mówiącą, że *istnieje metoda automatycznego przypisywania raportu o błędzie oprogramowania do odpowiedniej grupy badawczo-rozwojowej, odpowiedzialnej za rozwiązanie błędu, która przewyższa dobrze znane metody przypisywania raportów o błędach.*

Dokonano przeglądu literatury. Jako jeden z istotnych wniosków było zdefiniowanie luki badawczej wskazującej wady aktualnie używanych technik walidacyjnych oraz zaproponowanie alternatywnej pozbawionej wyżej wspomnianych negatywnych cech. Przeprowadzone eksperymenty przy użyciu protokołów wykorzystujących różne sposoby budowania zbiorów trenujących i testowych wykazały istotną różnicę w wynikach. Wykonano analizę możliwości zastosowania wyjaśnianej (wytłumaczalnej) sztucznej inteligencji.

Przeprowadzono szereg badań zarówno związanych z preprocessingiem, wektoryzacją danych źródłowych jak i na różnych poziomach struktury firmy. W ramach realizacji doktoratu zaproponowano nową metodę przypisywania raportów w kontekście kompozycji organizacji. Na postawie badań skuteczności wybranych algorytmów uczenia maszynowego, podjęto decyzję oraz wdrożono pilotażowe rozwiązanie w firmie. Ze względu na to, iż udało się zebrać w określonych warunkach zarówno predykcje modelu uczenia maszynowego jak i decyzje ludzi, zdecydowano się przeprowadzić analizę porównawczą wyników.

Badano możliwość zastosowania multimodalnych sieci neuronowych jak również innych sposobów rozwiązania problemu przypisywania raportów. Kluczowym aspektem było przygotowanie odpowiedniej reprezentacji danych wejściowych jak i zaprojektowanie wymaganej architektury multimodalnej sieci neuronowej. Przeprowadzone badania wykazały wyższość tej metody w porównaniu do zastosowanych metod referencyjnych.

Zaświadczenie o pracach wdrożeniowych w firmie zostało umieszczone w załączniku C. Zawiera ono informacje o trwających pracach związanych z wprowadzeniem multimodalnej sieci neuronowej. Pierwsze jej wersje są już wdrożone. Obecnie trwają prace związane z lepszym doborem cech i walką z brakującymi danymi. Ponadto zawarte są tam informacje na temat innych powiązanych wdrożeń związanych z raportami o błędach w oprogramowaniu.

ACKNOWLEDGEMENTS

With immense pleasure and deep sense of gratitude, I wish to express my sincere thanks to my supervisors Prof. Robert Burduk, DSc, PhD, Eng. and Michał Kucharzak PhD, Eng. from Wrocław University of Science and Technology, for their support, advice and encouragements.

I am grateful to NOKIA company for sharing essential data and providing me with infrastructural facilities and many other resources needed for my research. Last but not least, I would like to thank my coworkers for support in implementation.

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

Place: Wrocław

Date: 03/07/2024 **Łukasz Chmielowski**

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF TERMS AND ABBREVIATIONS

ACC - Accuracy
ARFCN - absolute radio frequency channel number
BTS - base transceiver station
CBOW - Continuous Bag of Words
CI / CD - Continuous Integration / Continuous Delivery
CNN - Convolutional Neural Network
DNN - deep neural network
EM - electromigration
EOS - Electrical Over Stress
ESD - Electrostatic Discharges
FN - False Negative
FP - False Positive
HCD - Hot Carrier Degradation
kNN - k - Nearest Neighbors
LDA - Latent Dirichlet Allocation
LIME - Local Interpretable Model-agnostic Explanations
LR - Logistic Regression
LSTM - Long Short-Term Memory
ML - Machine Learning
MOSFET - metal-oxide-semiconductor field-effect transistor
NB - Naive Bayes
REST API - Application Programming Interface
RF - Radio Frequency
SVC - Support Vector Classifier
TF-IDF - Term Frequency - Inverse Document Frequency
TN - True Negative
TP - True Positive
URL - Uniform Resource Locator [73]
XAI - Explainable Artificial Intelligence
XGBoost - eXtreme Gradient Boosting

CHAPTER 1

Introduction

1.1 Background

This paper concerns applications of artificial intelligence and machine learning in the context of applications related to software lifecycle processes [1]. Even though the quality of software [98] is sometimes marginalized in creating a variety of solutions for specific ones, it can be crucial. For high-responsibility and safety-related applications, the implementation of high-quality products is a key aspect. Unfortunately, even there problems cannot be avoided. For instance, a soviet satellite detected incoming missiles from the United States (in 1983). That warning was false. Fortunately, commanding officer decided to ignore it. In other applications where consequences are less hazardous but might have serious safety or financial implications. Time of solving a problem is key. For instance, customers of one of biggest US mobile phone operator AT & T, after a software upgrade in 1990, were not able to make a long distance call on that day. Losses valued at over 60 million dollars [54].

The author of this dissertation is personally involved in the development of Software Quality Assurance [110] tools in large scale telecommunication company. Some improvements were introduced into company internal systems even on a global scale.

1.2 Problem description, motivation, challenges

Software bugs and hardware defects are unavoidable in product development, such as base transceiver station (BTS). A software bug is an error, flaw or fault in a computer program or system that causes it to produce an incorrect or unexpected result, or to behave in unintended ways. The origin of the name of this term may be related to the following situation.

This bug (Figure 1.1), however, was literally a bug. "First actual case of bug being found," one of the team members wrote in the logbook. The team at Harvard University in Cambridge, Massachusetts, found that their computer, the Mark II, was delivering consistent errors. When they opened the computer's hardware, they found ... a moth.

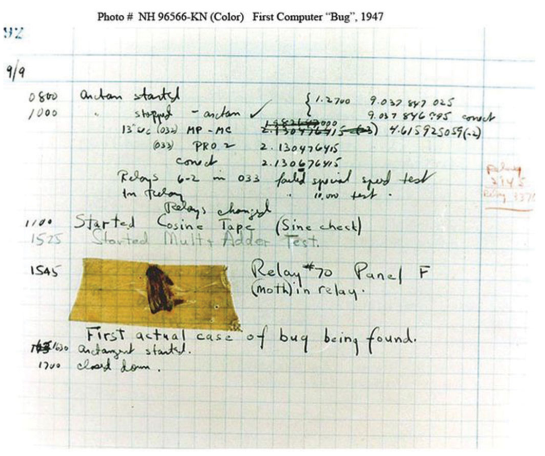


Photo # NH 96566-KN (Color) First Computer "Bug", 1947

92

9/9

0800 Arctan started
1000 " stopped - arctan ✓ {1.2700 9.037 847 025
9.037 846 995 conect
13" uc (033) MP - MC ~~1.982147000~~ ~~2.130476415~~ (~~23~~) 4.615925059(-2)
(033) PRO 2 2.130476415
conect 2.130676415
Relays 6-2 in 033 failed special speed test
in relay " 11.000 test .
Relays changed
1100 Started Cosine Tape (Sine check)
1525 Started Mult + Adder Test.
1545 Relay #70 Panel F
(moth) in relay.
First actual case of bug being found.
~~1600~~ 1630 Arctangent started.
1700 closed down.

Relay 2145
Relay 3370

Fig. 1.1 The First "Computer Bug" [104]

The trapped insect had disrupted the electronics of the computer [50].

Hardware bugs may be an effect of defects created in the design phase, during manufacture or operation of computer devices [124]. The problems of incorrect operation of system might be related to various reasons. Common ones are breakages of bound points, mechanical stresses, thermal change stresses, Electrostatic Discharges (ESD), Electrical Over Stress (EOS) and electromigration (EM) [76]. ESD may cause soft failures like temporary failure after which systems recovers after reboot or hard failures causing permanent damage of to the system [77]. Additionally, problems might be related to time-dependent wearout like Gate Oxide Wearout or Hot Carrier Degradation (HCD). Hot carriers degrade the gate dielectrics in MOSFET [113]. As a result of such processes shifts in threshold voltages may appears or even device failures [42].

To ensure the highest possible quality of the product, great emphasis is placed on its testing. A wide variety of tests are performed. Common fundamental ones include booting the system, upgrading, and downgrading software. A separate branch of testing is related to the functionality of the product, where the behavior of the system is verified against the specification. There are activities to check system performance, security, documentation, and scalability. Tests that are performed on a regular basis are called regression tests. Nevertheless, there is also another important category of robustness tests, which includes, e.g., power cycle tests, high availability tests, and special system configuration conditions with boundary values. As a result, software bug reports are created that contain discrepancies in the system's behavior [93].

Software testing can be divided into four different levels such as:

- unit testing,
- integration testing,
- system testing,

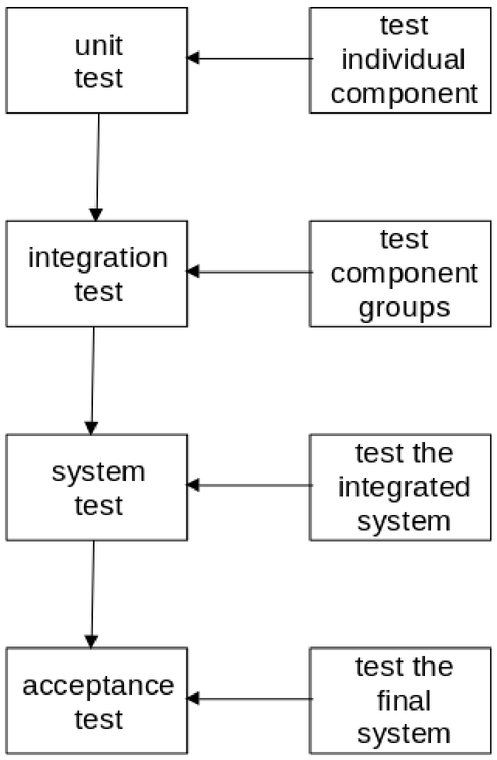


Fig. 1.2 Software testing levels [59]

- acceptance testing.

This is shown schematically in Figure 1.2. At unit testing level the smallest parts of code are being tested. Those parts of code are usually called units. At this step software testing, tests are generally prepared by the developers or workers which cooperate on a daily basics. Various scenarios describing interactions between components are being under verification during integration testing. System level testing checks if technical and functional requirements are met. Moreover, performance tests should be carried out at this step. Last but not least described phase is acceptance testing. Those tests may be performed at a software producer environment or directly at a customer one. At that stage actual user should be involved [59]. Referring to the problem of assigning a report of problem, we can expect that if software bug occurs at unit testing level it should be handled by one of developers responsible for development of this unit. More challenging part is when a software bug occurs at the later stage of development or even in real customer use. For large and complex systems even pointing out department can be a complicated task [45]. Departments are marked in Figure 1.4 as Group <letter><digit>, e.g., Group A1; whereas divisions are groups within departments marked as Group <letter><digit><letter><digit>, e.g., Group A1C1.

In fact, real applications employ continuous development. There can be implemented an approach like DevOps, see Figure 1.3. This methodology consists of two tasks at the same time. The origin of name comes from two terms Dev (development)

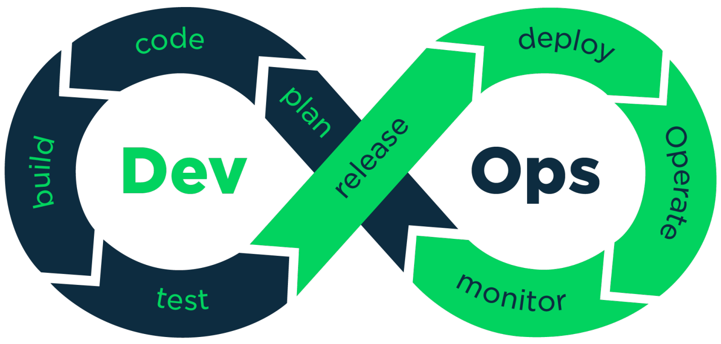


Fig. 1.3 Devops [45]

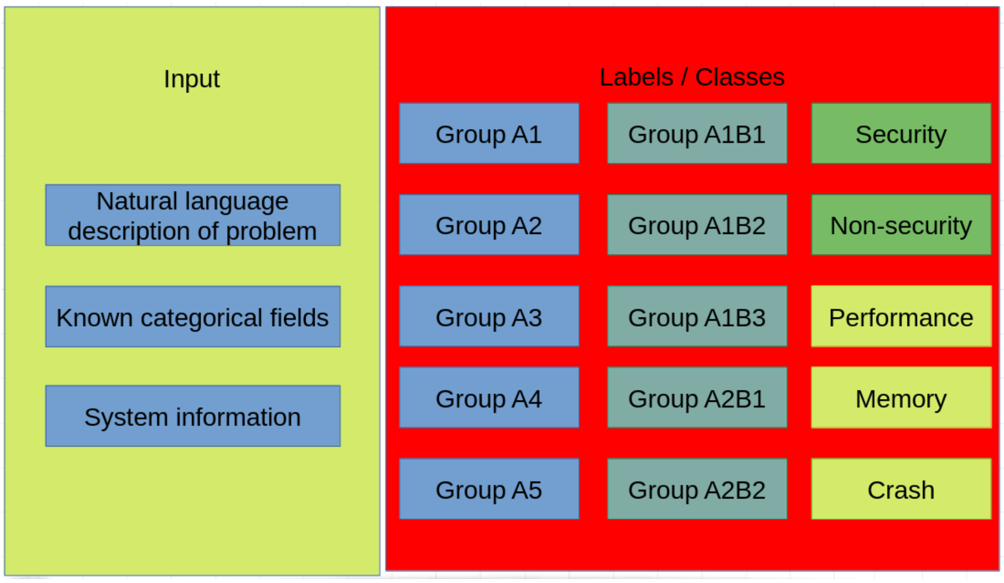


Fig. 1.4 Problem description

and Ops (operations). In short, DevOps is a set of practices which combines software development with operations that result in a better and faster software development cycle with high quality software and enables agile development. CI/CD is a principal part there. CI stands for continuous integration, whereas CD from continous delivery, or continuous deployment [46].

Discussed problem concerns about assigning a software bug report suggestions about to groups on different levels of organization which should be involved in investigation of problem or deliver correction. In addition, aspects related to whether the reports are related to security or not are considered, as well as an indication of the type of defect, such as related to, e.g., performance, memory or crash, see (Figure 1.4).

The business motivation to analyze the problem of assignment of bug report was the fact that enough good solution potentially may reduce time needed to assign a bug to the correct group which should be engaged in solving reported problems. Solving of reported problem may require delivery of source code patch, changes in documentation, analysis of environment. Sometimes enough is to tell that there is a need for little reconfiguration of setup or that the system works as expected by design, but still there is a need to assign to proper group of people who are able to deal with particular cases. It also might reduce number of unnecessary transfers between groups, but at the same time we must remember that the part of them is expected by design like transfer between group which investigate a problem and a group which implement the correction or if a correction is made during working hours in different time zones, then it might be reassigned to another group which works in a different time zone.

The specific aspects that make data analysis and implementation of the solution in the company difficult are related to:

- complexity of the system,
- experience of engineers submitting defect reports taking into account subjective perspective,
- the use of natural language by the reporters,
- a numerous of words/tokens which are non-English words,
- content of system logs which may vary depends on parameters of logging level respective for different components,
- variability of system logs resulting from development of product.

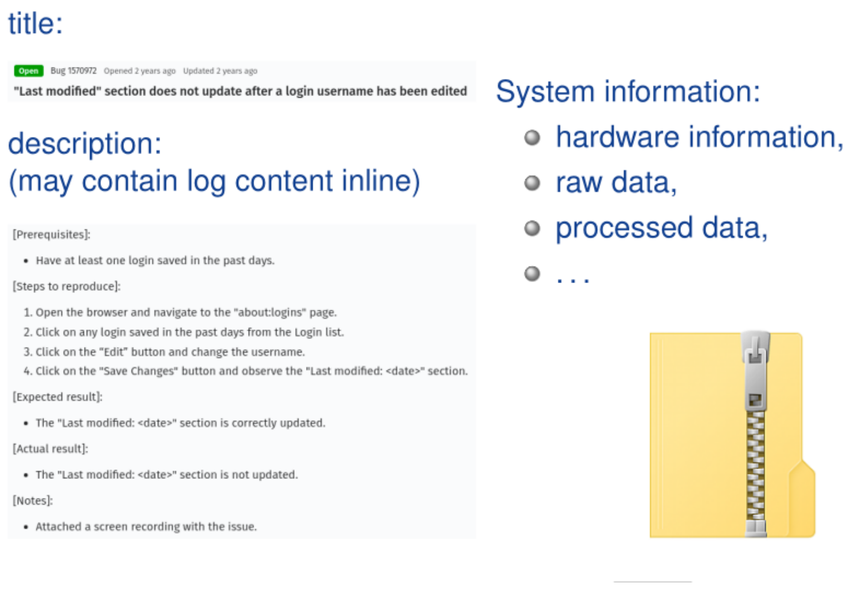
1.3 Description of single data record

Single record consists of among others title, description, detailed system logs, group or groups involved in solving problems unless it is in its first phase (Figure 1.5). System information as package of logs (snapshot) contains, among others, information about hardware used or log content. Snapshot may be transformed with usage of analyzers which are made by different groups of developers and highlight potential problems.

1.3.1 Natural language description of problem

Natural language description is always created within form with two separate fields. The first one consists of title of problem reports which might also include tags written as plain text at the beginning like '[TAG] Rest of title'. Second field is a short description of a problem, and it is written according to internal company template shown in listing A.1. It usually contains information like:

- expected result,
- actual result,
- detailed scenario of reproduction if applicable,



Open Bug 1570972 Opened 2 years ago Updated 2 years ago

"Last modified" section does not update after a login username has been edited

[Prerequisites]:

- Have at least one login saved in the past days.

[Steps to reproduce]:

1. Open the browser and navigate to the "about:logins" page.
2. Click on any login saved in the past days from the Login list.
3. Click on the "Edit" button and change the username.
4. Click on the "Save Changes" button and observe the "Last modified: <date>" section.

[Expected result]:

- The "Last modified: <date>" section is correctly updated.

[Actual result]:

- The "Last modified: <date>" section is not updated.

[Notes]:

- Attached a screen recording with the issue.

Fig. 1.5 Single data record

- execution log,
- . . .

This kind of document is called issue report, but is also known as anomaly report, bug report, defect report, error report, issue, problem report and trouble report, amongst other terms [60].

1.3.2 System information – raw log package and processed data

Log archive contains logs from main module (system module) of BTS and different connected devices like for instance radio modules. BTS consists of radio channels, each of which has its own absolute radio frequency channel number (ARFCN), as well as transmitter and receiver antennas. Telecommunications equipment is strategically placed on the tower to provide services in a specific geographic region. Strategic orientation of communication directional antennas affecting the quality of voice and data services provided to customers [20]. Log archive has a complex structure and contains a lot of various files and nested archives. Archives might be for instance related to corresponding radios. Inside its recent restarts of system module can be found with the reasons if known. There is a specific part of blackbox related to firmware where is placed serial number of main modules and data about recent updates of selected components.

Additionally, there may be available extracted data from log analyzers written based on experts' domain knowledge. The aim of these scripts is to get most important log messages and present its content to users with usage of graphical user interface, but one of novel approach was to use that information as part of input data to solve problem of assignment of bug report.

1.3.3 Additional remarks

The research experiments covered in this work are related to the usage of data in different forms. Firstly, possibilities of the use of representation with the use of natural language form and categorical data were evaluated. Next solutions which use extracted data from log analyzers were involved. Finally, a solution has been applied, which used all three of mentioned type of data. That data come from Nokia company. Please note that additional other data from Mozilla Dataset were utilized in studies in Chapter 5.

1.4 Scientific motivation and research gap

To the best of author's knowledge, there is no application with usage of multimodal neural networks with combined information from text description and system information like for example hardware used or analyzed log content. This thesis is related to novel application of multimodal neural networks in problem of assigning group for handling a bug. There is a possibility to try to use multimodal deep learning approach to solve the presented problem. Multimodal deep learning is typically used in problems where we have combined data from audio and video [95]. Another currently used application of deep multimodal network is an application where pictures and text are combined [66, 79]. Multimodal representation learning is a type of representation learning, which learns features from multiple modalities, and these should not be independent. Correlations and associations among modalities should be present [134]. In case of usage information about hardware it might give potential profits in case of bugs related to specific radio modules or hardware, but usage of analyzers might have positive impact in general. To potentially introduce multimodal neural network there is a need to verify possibilities of applying that in the context of software bug report assignment. There is a need to design the exact architecture of the network, conduct research on possibilities of representation of data and choose input types for the network. Part of them is related to representation of data based on natural language processing, part related to title and description, part related to categorical data, another part is related to representation of available system information. To apply methods related to multimodal learning or multimodal neural networks are used both of those representations. Overall, there are also considered research gaps strictly related to the subject of assignments of software bug reports like possible industry applications, methods of evaluation of models created specifically for that purpose. The impact of time dependencies related to time of the creation and resolving the issue on results obtained with the use of machine learning models were investigated. For industrial applications also was evaluated the possibility of use of XAI methods. There was a need to inspect the influence of applying algorithms or heuristics related to XAI methods on results and also verify whether the results of explanations seem to be consistent for domain experts.

1.5 Research questions, thesis, its aims and goals

Research aims and goals

One of the work's goals is to review solutions in areas related to the applications of artificial intelligence or machine learning solutions in handling software bug reports. The aim of the dissertation is to verify the possibility of using a multimodal neural network for that purpose. For this verification, it is necessary to propose the architecture of such a network and to design a representation of the data provided as input. However, this is not the only method that falls under the scope of this work, as there was a need to review modern solutions in this field. It is also important to propose the implementation of system solutions in the company if satisfactory results are achieved.

Thesis

There exists a method for automated assignment of a software bug report to appropriate development group, responsible for resolving the bug, which outperforms well-known methods for bug report assignment.

Research questions

This dissertation discusses the following research questions:

RQ.A are related to predictions on department level

- RQ.A.1: What is an impact of stemming and lemmatization on bug assignment accuracy?
- RQ.A.2: What is an impact of different settings of n-gram and max features of TF-IDF on bug assignment accuracy?
- RQ.A.3: What is an impact of additional information from bug report related to product type on bug assignment accuracy?

RQ.B are related to predictions on division level

- RQ.B.1: What is an impact of introducing multimodal neural network on bug assignment accuracy?

RQ.C are related to state of art of current evaluation

- RQ.C.1: Are the standard machine learning methods for evaluation are appropriate to evaluate problems related to bug handling?
- RQ.C.2: If not then what experimental protocol should be introduced?

RQ.D: comparison results of machine learning predictions versus humans' predictions?

- RQ.D.1: What are results of human predictions on department and inside department level?
- RQ.D.2: What is the relation between results of machine learning predictions versus human predictions?
- RQ.D.3: What are the factors impacting human predictions?

RQ.E what are benefits of proposed solution?

- RQ.E.1: What are minimal requirements for solution applicability in software development company?
- RQ.E.2: What are the scenarios of deployment of application?
- RQ.E.3: What are main advantages and disadvantages against humans?

RQ.F potential application of XAI

- RQ.F.1: What is accuracy when comprising standard and easily explainable algorithms?
- RQ.F.2: What benefits that might gives?
- RQ.F.3 Does information provided by explainable models seems to be consistent?

1.6 Planned research methods

In purpose to answer research questions (RQ.A) which are related to predictions of assignments on department level especially about an impact of stemming, lemmatization, different settings of n-gram and max features of TF-IDF, additional information from bug report related to product type and algorithm selection on bug assignment accuracy there was a plan to use numerical simulations. Data to perform almost all of simulations were gathered from historical reports of problems inside NOKIA company. For those data it is planned to split them into training and testing set. An evaluation was planned to be done on newer data than training one. A similar experimental protocol was planned to be introduced during work related to predictions inside the department (RQ.B). There was a need to verify an impact of introducing multimodal neural network on bug assignment accuracy. As input related to natural language part was planned to utilize best settings obtained during performing experiments on depatment level. For the reference it was planned to utilize neural network similar in build to logistic regression. There was performed exploratory research related to state of the art of current methods of evaluation related to solving problems. Analytical research was planned to be introduced to compare results of human decisions and machine learning predictions. What is more, it is planned to find out whether the comparison according to simple check of first human prediction is adequate? Conclusive research aims at potential benefits of proposed implementation of solution of automated assignments of reports of problems? What are the minimal requirements to introduce them. What can give specific postprocessing method with thresholds? What is the main advantage and disadvantage comparing to humans? Descriptive research was planned to be applied for considering potential application of XAI to extract knowledge based on XAI analysis. What is more there were considerations whether it is possible to extract knowledge to classify whether bug is related to security or not?

1.7 State-of-the-art

1.7.1 Papers related to the topic of software bug report assignment

There are already present publications related to software bug reports. One of them, [127], shows a solution to predict a faulty component. It uses call stack, primary data of bug reports in that work. Bug handling process demonstrated there include both assignment and reassignment of issue in case if wrong component was previously indicated. In that solution as a machine learning model were chosen finally random forest with parameters tuned by GridSearchCV, [101]. However, there were considered and verified different solutions such as decision tree [48], LSTM [58], feedforward neural network [25]. Before feeding these data into the model, there were previously applied preprocessing and extracting component features. Another Paper [135] utilizes Latent Dirichlet Allocation (LDA) [27]. Then the representation is transformed to a feature vector and then fed into Deep Neural Network.

In the literature it is mentioned not only assigning the components, but also different tasks. There is even a term bug triage in the nomenclature. This expression is used among others in [83, 131, 132]. The problem related to duplicates of software bugs are mentioned in, e.g., [69, 82, 83]. Paper [117] discusses a possibility of detecting bug report duplication with the use of Support Vector Machine to solve that problem.

There are created pairs from duplicated reports as well as there are generated pairs of non-duplicated cases. To fit model outputs were assigned for given pairs as follows, for duplicate pair was given 1, whereas for non-duplicate set -1. No all possible combinations of pairs were taken due to fact that there would be significant imbalance of type of pair in training set. In Mozilla Project were introduced a solution called just-in-time retrieval. Authors of [126] claims that even after introduction of that, there are still over 10% of bug duplicates. Assignment of priority might also be important to be created automatically, especially as it might be expected early response for those cases. Paper [26] utilizes TF-IDF and Naive Bayes, [122], for identification of security and non-security cases. Paper [94] related to software bug assignment uses both natural language description and discrete features. Principal Component Analysis is performed on one-hot encoded data. Whereas, on textual data natural language processing techniques are used to tokenize and then entropy-based feature selection, [114], is applied.

There are multiple papers which utilize neural networks for tasks related to software bug report assignment. In [86] is applied Bidirectional Long Short-Term Memory Network was considered to automate the bug triaging process in publication.

Predicting simultaneously developer and the team is introduced in [41], where is used two-output deep neural network architecture (Dual DNN). The used network is designed as follows. There is an input layer where data are fed which are previously transformed with the use of latent semantic analysis, [72]. After that are two hidden layers and two output layers. The first output layer is related to developers, the other is related to teams. There exists one more connection designed in that way that, the output of team layer is fed into developer layer.

1.7.2 Natural language processing

Natural language processing is an essential part of work as problem reports contain descriptions in natural language. Processing of natural language may contain the following steps preprocessing (see Section 2.3.2), vectorization (see Section 2.3.3). Preprocessing is a phase where the text is being prepared for further steps. Raw data are cleared, for instance, can be there removed special characters, letters converted to lower case, performed stemming or lemmatization. Following that step to provide data to machine learning models, there is a need for change the representation of text data to numerical representation. Example of methods are bag of words, term frequency, term frequency – inverse document frequency, continuous bag of words, [88], skipgram, [88]. Having such representation can be applied in the next step of processing like for instance clusterization, classification, or labeling. Then is performed the task, which is required in specific problems, e.g., clusterization, classification, labelling.

1.7.3 Validation techniques

For creation of train and test set usually in the context of evaluation of machine learning models there is a need for selecting data for train and test sets. That might be done randomly, or with the use of split by time. There might also be an applied stratification process which tries to ensure that the data is split such as that each part contains similar number of instances related to given class in each set or fold. A fold is a set created by Cross-Validation method. In k-fold cross-validation the data is divided into k folds of equal size (or nearly equal). For each iteration of algorithm, the data from k-1 folds are taken as a train set, and the one which is not present in train set is used as validation fold. The predictions of the model are made on validation fold in each iteration. Then the results are summarized. Different methods might be used to achieve aggregated measures from these samples, like for instance average [106].

Techniques like cross-validation are utilized for evaluation of software bug reports related tasks. For instance, in paper, [70], related to predicting whether the first assignment of bug report is likely to be reassigned in the future and in [14] where bug reports are triaged with the use of Latent Semantic Indexing and Support Vector Machine.

In the context of tasks related to software bug reports assignment there was a need to introduce the solution which includes not only splitting by date, but also the fact that from reporting software malfunction to resolving the issue, time passes. That fact was not taken for evaluation in the context of that specific problem.

1.7.4 Usage of thesholds

Machine learning models may have a possibility to return not only chosen decision by model, but also that result may be supported by probability estimates. That fact has already been applied in solutions related to software bug reports transfers in that fact that only ones with the highest probability are transferred [112]. Because there is a compromise between the accuracy of the solution and the number of requests handled, the golden mean must be established somewhere. The split point can be called threshold or cutoff value.

There was lack of solution which enables the usage of such thresholds in the context of organizational structure. One of the possibilities of such an application is transferring bug reports based on outputs from models which predict groups on distinct levels of organization. Such a solution also may have final decision only based on thresholds.

1.7.5 Usage of explainable artificial intelligence

Explainable artificial intelligence allows solutions of machine learning to be explainable to the end user. It might be used for validation of models before production or to explain given decisions. That might be utilized as a decision support system, so the decision is made by a person who may utilize the prediction of model as well as explanations if given. In practical use, the decision might be made automatically with a machine learning model and recorded with explanation. That might be utilized later in case of doubts of such decision. Solutions with the usage of recommendations have already been introduced in software development. There is an application related to prediction which delivers explanations about chances of introducing bug within the software commit [65].

There were not found the works using explainable techniques in the context of software bug assignment.

1.7.6 Multimodal neural network

Multimodal neural networks are usually applied in the context of applications related to videos where one modality comes from audio whereas the second from image. Those applications might be related to emotion recognition. An example is related to work, whereas one input to neural network is related to vocal embedding, and the second related to visual embedding [34]. The predicted class comes from a set of emotions like, e.g., neutral, happy, sad, surprised.

Currently many blogs and posts are supplemented with hashtags. To suggest one appropriate might also be used multimodal neural network. Encouraging results were obtained when as input was taken image and text in [128].

In [74] a solution is shown where input data come from camera to capture vision, and sensors related to the position of robotic arm, and force sensor. In network is created multimodal representation. The goal of that work was related to evaluation the ability to transfer multimodal representations across tasks and the value of combination of multisensory information.

Another field of application where multimodal neural networks are utilized is the use for Geosciences. An example related is to classification of flood tweets. There were used not only textual features come from messages, but also information related to location and time of publishing the content [44].

Multimodal neural network has been applied for the solving of task of estimation of energy and time usage in 3D Printing [35]. As inputs was taken information related to time series extracted from G-code files, [67], and images extracted from 3D digital model. The results showed the effectiveness of that approach on real-world data.

On the global market there is a lot of mobile software. For the task of categorization Two-Phase Multimodal Neural Network was used. Inputs come from applications packages and are related to titles, permissions and file named Strings.xml. Then various representations are utilized followed by separated parts of neural networks, then concatenated, and fed to the next part of the neural network [108].

Multimodal Neural Networks have also been utilized for public opinion risk monitoring and early warning systems. The collected data like images and text come from social media [125].

To utilize Multimodal Neural Network in the context of software bug report assignment with the use of both natural language form, categorical fields, or content extracted from system information there is a need to create a representation for input layers and create specific architecture of neural network which would be appropriate for solving that problem.

Topics related to state-of-the-art are also covered in Sections 2.1, 3.2, 4.1.3, 5.1 and 6.1.

1.8 Content of document

1.8.1 Overview

Chapter 1 contains introduction, motivation, challenges, and general description of data used for research. There is covered general knowledge about state-of-the-art, research methods and questions. Following that are present modified articles, detailed information about them is in Section 1.8.2. Chapter 7 describes part of work related to environment and orchestration. Summary of that work and information about future works are placed in Chapter 8. Appendix A contains template of report of bug, whereas Appendix B covers user guide for accessing machine learning based bug assignment predictions. Appendix C include attestation of implementation of selected parts in company. It describes what author of dissertation introduced in the company systems and what methods are still ongoing related to works which are taken.

1.8.2 Scientific articles included in dissertation

- Chapter 2 is a modified article about the impact of software bug report preprocessing and vectorization on bug assignment accuracy. It covers general knowledge about bug assignment, description of typical natural language pipeline and research questions related to that part of work.

  L. Chmielowski and M. Kucharzak. Impact of software bug report preprocessing and vectorization on bug assignment accuracy. In M. Choraś, R. S. Choraś, M. Kurzyński, P. Trajdos, J. Pejaś, and T. Hyla, editors, Progress in Image Processing, Pattern Recognition and Communication Systems, pages 153–162, Cham, 2022. Springer International Publishing. [38]

  Contribution of author of dissertation:

  Conceptualization; Methodology; Software; Writing - Original Draft independently conceived the experiments; analyzed results; wrote original draft; provided editorial suggestions; conducted editing of work; attempted to disprove the novelty; reviewed the manuscript

- Information about predictions both at department level and division level is covered in Chapter 3. There is also present material containing benefits of proposed solution based on specific combination of results from machine learning models where at least one predicts department and at least one predicts division.

  L. Chmielowski, P. Konstantynov, R. Luczak, M. Kucharzak, and R. Burduk. A novel method for software bug report assignment. Reliability Theory and Applications. [37]

  Contribution of author of dissertation:

  Conceptualization; Methodology; Software; Writing - Original Draft

  independently conceived the experiments; analyzed results; wrote original draft; provided editorial suggestions; conducted editing of work; attempted to disprove the novelty; reviewed the manuscript

- Chapter 4 is related to advances in problems of evaluation of software bug report assignment.

  L. Chmielowski, M. Kucharzak, and R. Burduk. Novel method of building train and test sets for evaluation of machine learning models related to software bugs assignment. Scientific Reports, 13, 12 2023. [40]

  Contribution of author of dissertation:

  independently conceived the experiments; analyzed results; wrote original draft; provided editorial suggestions; conducted editing of work; attempted to disprove the novelty; reviewed the manuscript.

- Chapter 5 show potential applications of explainable artificial intelligence in bug assignment in software development.

  L. Chmielowski, M. Kucharzak, and R. Burduk. Application of explainable ar tificial intelligence in software bug classification. Informatyka, Automatyka, Pomiary w Gospodarce i Ochronie Srodowiska, 13(1):14–17, Mar. 2023. [39]

  Contribution of author of dissertation:

  independently conceived the experiments; analyzed results; wrote original draft; provided editorial suggestions; conducted editing of work; attempted to disprove the novelty; reviewed the manuscript

- Chapter 6 is related to potential application of multimodal neural networks in solving problem of labeling bug reports.

L. Chmielowski, P. Konstantynov, R. Luczak, M. Kucharzak, and R. Burduk. Application of multimodal neural networks in solving problem of labeling bug reports. [36]

Contribution of author of dissertation:

Conceptualization; Methodology; Software; Writing - Original Draft

independently conceived the experiments; analyzed results; wrote original draft; provided editorial suggestions; conducted editing of work; attempted to disprove the novelty; reviewed the manuscript

CHAPTER 2

Impact of software bug report preprocessing and vectorization on bug assignment accuracy

This chapter is modified from Impact of Software Bug Report Preprocessing and Vectorization on Bug Assignment Accuracy in collaboration with Michal Kucharzak [38].

In modern and professional large software development organizations, bug handling processes are an important part of the software lifecycle and usually have a very formal process definition. To effectively provide resolution, improvements to such a process may include automatic bug assignment, which is a task of selecting the proper team of developers to further investigate the bug report. This research focuses on the impact of natural language preprocessing and vectorization on the accuracy of assigning error reports based on data captured in large software development projects. The results of the experiments include stemming and lemmatization techniques used to pre-process the description of bug reports and parameterization of term frequency – inverse document frequency (TF-IDF).

2.1 Introduction

An integral part of the software lifecycle is software testing. It becomes more resource-intensive as the complexity of the system or product increases. Bugs or errors can occur during software testing processes, and their proper operation is a key task to meet time-to-market requirements as well as improve software quality. Assigning a bug report to appropriate developers or development teams for further maintenance is known as "bug triage" [131, 132], which may also involve assigning priority [15, 135] to issues. To manage software bugs in the practice of software development, bug tracking systems are widely used [78]. Such systems are used by users to report problems or bugs in an organized and process-defined manner. Thanks to that, bug report repositories can be later utilized for data mining.

This paper focuses on the automatic assignment of a software bug to the proper development team, based on the data in the software bug report. Such a report typically includes a title and description in natural and free-form language, as well as additional data or predefined parameters from a limited set of options, such as software version or product type. A natural language title most likely contains a short summary of the observed issue, while a description provides more detailed information about the problem. The description should include specific information for developers and other parties on any adverse events that have occurred, e.g., the stages of the test necessary to reproduce the fault, requirements not met, comparison of actual and expected results with justification, references to historical results, etc.

The quality and accuracy of the software bug report description depends largely on the system's complexity in large development projects. The same effect can be noticed by different tests in different contexts, and the description of the fault often varies depending on the test engineer's subjective perception, grammar, typos, language style, or observation correctness. It becomes a more complex task which requires natural language processing research to implement automated tools assisted by machine learning techniques. Moreover, accurately assigning a software fault report based on natural language descriptions becomes more challenging. The impact of natural language preprocessing and vectorization methods on software bug report assignment on the accuracy of software bug assignment has been inspected.

Numerical experiments based on real data from a large software development organization are the main research contribution of this work. Verification teams located in many different countries create software bug reports in English, what also affects the natural language style used in the documents. The bug report repository contains descriptions of numerous bugs reported.

2.2 Related works

Text fields in fault reports typically contain text in free form, including software system log printouts, source code, specific technical words, or hashes. Many papers related to software bug reports discuss the topic related to natural language processing. The paper [137] considers software-specific recognition of named entities using a linear string of conditional random fields. A machine learning tool that automatically assigns a product and component for every incoming Mozilla Firefox bug based among others on texts in free form utilize the xgboost classifier [81]. The solution utilizes processing of the natural language pipeline title and comments on the software bug report.

The solution to predict the reassignment of fault reports based on data from different Eclipse and Mozilla products with the use of removing stopwords, stemming, and Naive Bayes classifier was described in [70]. Bayesian approach has also been utilized in [43] for software bug report assignments. Paper [55] shows an analysis of the possibility of predicting the severity level of fault reports with the use of the *term frequency –*

inverse document frequency (TF-IDF) and the Naive Bayes classifier. There was also a proposal for the creation of a dictionary containing critical terms, which might be useful for assigning the severity level.

What's more, there are also papers related to research about possible applications of machine learning solutions to determine whether a bug is security-related or not [26, 53]. Another article [103] which was created by Microsoft on fault security classification includes three techniques: boosted decision trees, logistic regression, and Naive Bayes, logistic regression. Experts from the University of Toronto and Intel introduced Dual-output deep neural network for automated fault classification at the team and developer level in [41]. At Ericsson, a doctoral dissertation related to automatic fault assignment and automatic fault location was conducted [62]. Paper [117] shows Support Vector Machine used to find duplicate fault reports. The problem of qualifying tickets as bugs, documentation requests, refactoring requests, improvement requests, feature requests, or others was discussed in [63] where *fastText* was utilized in order to perform that task.

The article focuses on the comparison of the effects of preprocessing (stemming and lemmatization techniques) and vectorization (TF-IDF) on the accuracy of software bug assignment using logistic regression.

2.3 Natural language processing pipeline

2.3.1 Typical natural language processing pipeline

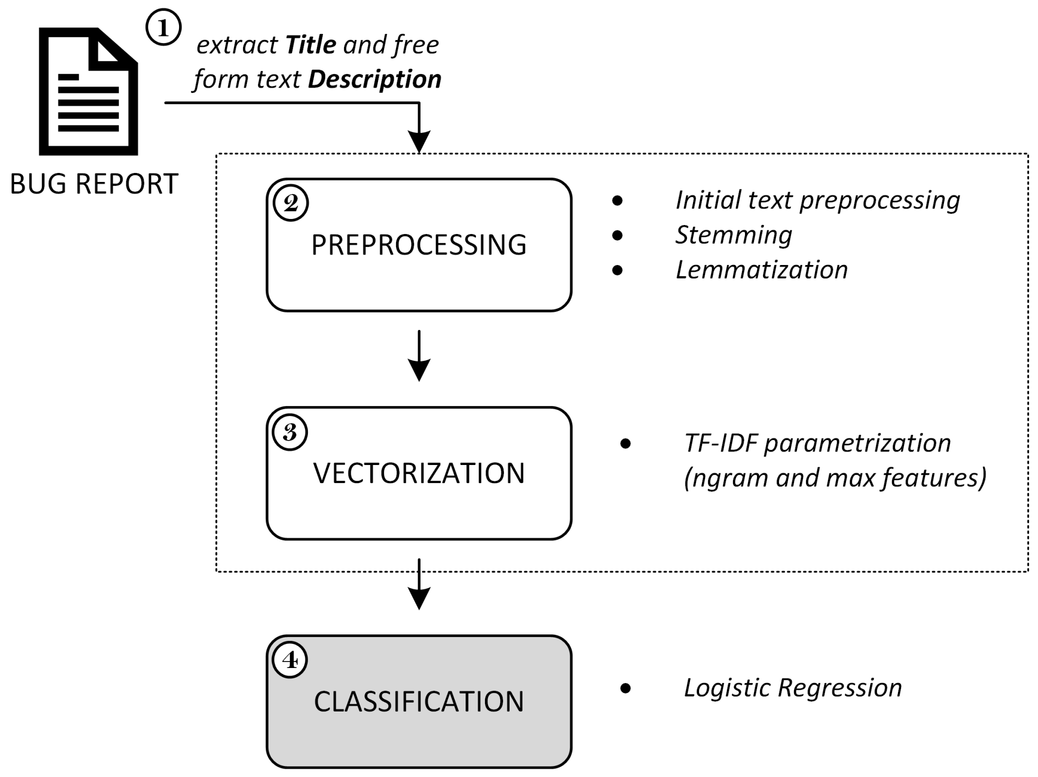
A typical natural language pipeline may have a couple of stages, like preprocessing, vectorization, and finally clustering, labeling, or classification. The main goal of preprocessing is to modify the raw text, for instance by inflectional suffixes, or special characters. Creation of the representation of text data as a vector of numeric data is called vectorization. Finally, the numeric data is utilized by classifiers.

Figure 2.1 presents a general diagram of automatic software bug report assignment, among others, a typical natural language pipeline, described later. Title and description fields ① as input are taken directly from the fault report.

2.3.2 Preprocessing methods

Preprocessing stage ② (see Figure 2.1) modifies the natural language-based title and description for next stages by performing selected operations:

1. change the letters to lowercase,
2. remove special characters,
3. remove stop words.



① extract **Title** and free form text **Description**
BUG REPORT
② PREPROCESSING
Initial text preprocessing
Stemming
Lemmatization
③ VECTORIZATION
TF-IDF parametrization (ngram and max features)
④ CLASSIFICATION
Logistic Regression

Fig. 2.1 Flow chart presenting pipeline related to machine learning solution of software bug assignment including among others preprocessing and vectorization.

Table 2.1 Sample cases of stemming and lemmatization

operation used	Example 1	Example 2
None	package is missing at receiver	connection has been established
Stemming	packag is miss at receiv	connect ha been establish
Lemmatization	package be miss receiver	connection have be establish

Common standard methods were used, such as changing the letter to a lowercase letter (step 1). In step 2, special characters (for example, `$`, `#`, etc.) are removed, leaving only alphanumeric characters and an underscore. Then, in step 3, stop words such as `the`, `or` are removed.

Stemming or lemmatization was utilized in the preprocessing phase, depending on the type of the experiment. The goal of stemming processes is to reduce the inflection of words to their root forms. Stem does not have to be the correct word. Root is a canonical form of lemmatization [61]. Table 2.1 presents examples of stemming and lemmatization.

2.3.3 Vectorization methods

Because numerical representation is a required input into algorithms, vectorization ③ is used to transform natural language text into its vector numerical form. Common numerical representations are bag of words (multiset words), binary frequency (set of words), term frequency (TF), or term frequency - inverse frequency of the document (TF-IDF) [107]. The term can be interpreted as one or more words. In addition, texts in natural language can be represented by n-gram sequences, i.e., consecutive positions of neighbors n (e.g., words) are treated as a single term. For simplicity, a single term defines a single feature. The equation (2.1) presents TF representation, where d is a document (e.g., fault report) and t is a term. The formula (2.2) shows TF-IDF representation. The equation (2.3) defines the IDF, where P means the corpus, for instance, the set of all documents d. Moreover, the space of terms can be constrained by an additional vectorization parameter called `max features`, which sets the maximum number of features used. Eventually, after preprocessing and vectorization, the classification ④ of the bug report is made utilizing Logistic Regression.

$$\text{TF}_{t,d} = \frac{\text{no. of occurrence of } t \in d}{\text{total length of } d} \tag{2.1}$$

$$\text{TF-IDF}_{t,d} = \text{TF}_{t,d} \times \text{IDF}_t \tag{2.2}$$

$$\text{IDF}_t = log_{10} \frac{\text{no. of documents } \in P}{\text{no. of documents of } P \text{ containing term } t} \tag{2.3}$$

2.4 Numerical experiments

2.4.1 Research questions

The aim of numerical experiments is to evaluate preprocessing techniques and parameterization of TF-IDF vectorization. Below are research questions presented:

- RQ.A.1: What is an impact of stemming and lemmatization on bug assignment accuracy?
- RQ.A.2: What is an impact of different settings of n-gram and max features of TF-IDF on bug assignment accuracy?
- RQ.A.3: What is an impact of additional information from bug report related to product type on bug assignment accuracy?

2.4.2 Description and experimental protocol

Due to trade secrets, it was not possible to provide details on the number of samples, but information on data distribution and division into train and test set is presented in Table 2.2. The selected departments are the five biggest in the company. The training set

Table 2.2 Train and test set data distribution

department name	percent of reports from training set	percent of reports from test set
Department A	27.3 %	19.7 %
Department B	25.7 %	32.5 %
Department C	16.4 %	17.5 %
Department D	15.8 %	13.5 %
Department E	14.9 %	16.9 %

comprises data for 12 months, while the test set contains newer fault reports collected over the next 1 month. Many technical phrases, abbreviations, as well as software-specific configuration names, values, or parameters are comprised in software bug report titles and descriptions. The accuracy of the bug assignment is further verified using Logistic Regression with the use of the same train and test sets for all calculations that were carried out. Text preprocessing, a common step in all experiments, is described in section 2.3.2. The following metrics are calculated: accuracy, precision, recall, F1. Reasons for choosing indicators:

- accuracy is a fundamental global metric in classification problems,
- precision is a fundamental metric in information retrievals systems,
- recall that determines sensitivity,
- F1 is an additional balanced metric between precision and recall.

2.4.3 Results and lessons learned

Experiment results showing accuracy, weighted precision, weighted recall, weighted F1 measures for various preprocessing options are shown in Table 2.3. Basic preprocessing is compared to one with application of stemming and lemmatization. In addition to title and description in natural language form, product type was utilized as an additional feature. Accuracy of software bug report assignment increases when using the product type, by about 1.0%, 0.8% and 0.7%, respectively, for lemmatization, stemming and without these techniques. What is more, even without the use of the product from the software bug report reports, lemmatization gives better results than with the use of stemming, which exceeds the basic preprocessing by 1%.

Table 2.4 and Figure 2.2 present the effect of TF-IDF parameterization vectorization on bug assignment accuracy. Experiments with settings n-gram range: 1—3, max features: 32k; n-gram range 1—2 and max features 64K led to the best results. In general, increasing the maximum number of features provides an improvement in accuracy; after the value 32k is reached, the results stabilize, and no additional enhancement is observed. Up to 16k as the maximum number of features in TF—IDF, introducing bi-

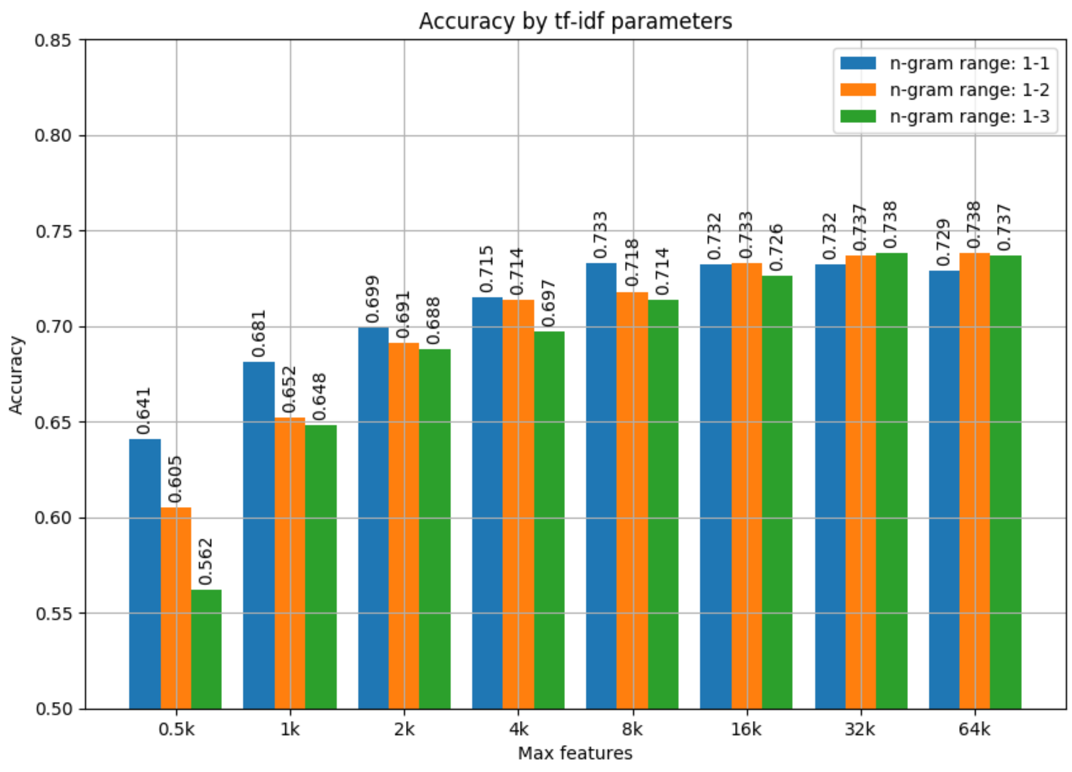


Fig. 2.2 Accuracy in function of TF-IDF parameters

Table 2.3 Impact of changing method of preprocessing on results

Preprocessing option	acc	precision	recall	F1
lemma	0.728	0.742	0.728	0.728
lemma (+product)	0.735	0.749	0.735	0.736
stem	0.722	0.733	0.722	0.723
stem (+product)	0.728	0.741	0.728	0.730
none	0.720	0.732	0.720	0.722
none (+product)	0.725	0.737	0.720	0.722

grams and trigrams usually positively impact the results. For bigger values like 32k or 64k, using n-grams may still improve the quality of outcomes.

Table 2.5 presents results of calculations where the n-gram range was set to 1—3, the maximum number of features to 32k and lemmatization and product information were utilized. Results are much worse for one Department (A) compared to the others. This may be due to the fact that keywords (terms) in that case are not so specific, or due to differences in distribution in train and test sets, see Table 2.2.

2.5 Summary

Assigning software bug reports to the department in charge of reported issues and further handling of the use of machine learning solutions is considered in this article. Var-

Table 2.4 Impact of n–gram range and maximum number of features set on results

Vectorizing option	acc	precision	recall	F1
max features: 0.5k, n-gram range: 1–1	0.641	0.656	0.641	0.64
max features: 0.5k, n-gram range: 1–2	0.605	0.62	0.605	0.604
max features: 0.5k, n-gram range: 1–3	0.562	0.579	0.562	0.56
max features: 1k, n-gram range: 1–1	0.681	0.703	0.681	0.685
max features: 1k, n-gram range: 1–2	0.652	0.668	0.652	0.653
max features: 1k, n-gram range: 1–3	0.648	0.663	0.648	0.648
max features: 2k, n-gram range: 1–1	0.699	0.716	0.699	0.701
max features: 2k, n-gram range: 1–2	0.691	0.709	0.691	0.693
max features: 2k, n-gram range: 1–3	0.688	0.705	0.688	0.689
max features: 4k, n-gram range: 1–1	0.715	0.73	0.715	0.717
max features: 4k, n-gram range: 1–2	0.714	0.729	0.714	0.715
max features: 4k, n-gram range: 1–3	0.697	0.712	0.697	0.699
max features: 8k, n-gram range: 1–1	0.733	0.747	0.733	0.735
max features: 8k, n-gram range: 1–2	0.718	0.732	0.718	0.719
max features: 8k, n-gram range: 1–3	0.714	0.731	0.714	0.716
max features: 16k, n-gram range: 1–1	0.732	0.745	0.732	0.734
max features: 16k, n-gram range: 1–2	0.733	0.746	0.733	0.735
max features: 16k, n-gram range: 1–3	0.726	0.74	0.726	0.727
max features: 32k, n-gram range: 1–1	0.732	0.746	0.732	0.734
max features: 32k, n-gram range: 1–2	0.737	0.751	0.737	0.739
max features: 32k, n-gram range: 1–3	0.738	0.753	0.738	0.739
max features: 64k, n-gram range: 1–1	0.729	0.743	0.729	0.73
max features: 64k, n-gram range: 1–2	0.738	0.753	0.738	0.739
max features: 64k, n-gram range: 1–3	0.737	0.751	0.737	0.738

ious preprocessing and vectorization settings were inspected. For comparing obtained results, metrics like accuracy, weighted precision, weighted call, weighted F1 were used. There are no major improvements in the change between stemming and lemmatization, however applying lemmatization and an extra feature related to product type slightly enhances results. Nonetheless, obtained results with the use of preprocessing methods are still better than those from initial text preprocessing. An increase in accuracy is visible when bigrams and trigrams are utilized compared to the situation when only unigrams are utilized as terms in the TD—IDF vectorization. Enlarging the maximum number of features used enhances the results to some value of that parameter set. It stabilizes at about 32k, further increasing that value might not have a positive effect.

Future work will include research which utilizes other vectorization techniques like *Skip-Gram*, *fastText*, *CBOW* [89]. There is also a plan to use different classification methods which will take as an input preprocessed and vectorized data related to software bug reports.

Table 2.5 Results for calculations with following settings: n-gram range: 1–3, maximum number of features: 32k.

Department	precision	recall	F1
Department A	0.560	0.711	0.627
Department B	0.787	0.864	0.824
Department C	0.759	0.683	0.719
Department D	0.829	0.629	0.716
Department E	0.846	0.671	0.749

CHAPTER 3

A novel method for software bug report assignment

During the development of software and electronic devices, it is inevitable to make mistakes. In large, developed companies, assigning a request to the right development team or even a department is not an easy task. Often, the creation of software bug reports and assignment to groups is also formalized by appropriate processes. The paper presents a novel method of software bug report assignment to a group of developers or analysts. A specific usage of organizational structure at the company is a key component of the proposed approach. There are presented results from real use application including both machine learning predictions and human decisions. Human predictions are not independent, the issues are raised as to why comparing the results of machine learning models with those of humans may be inappropriate and what factors influence human decisions. The work also covers conclusive research about potential benefits of the application of automated assignment of bug reports.

3.1 Introduction

Discussed problem concerns about assigning a software bug report to correct group automatically based on given data like description, system information in raw format or already processed by analyzing tools. Approaches similar to these presented in the paper may be applied to situations which work with other software development cases like related to feature requests, supporting questions or similar issues which should be handled during software development or maintenance. The approach might also be applied to any other different task related to machine learning tasks like classification or labeling in a similar context. It is expected that if a software bug occurs at unit testing level it should be handled by one of developers responsible for development of this unit. More challenging part is when a software bug occurs at the later stage of development or even in real customer use. For large and complex systems even pointing out department can be a complicated task [45]. Additionally, there is an assumption that the company is divided into at least two organization levels, like departments and divisions, as shown in Figure 3.1. Please note that the names "department" and "division" and relations between them are shown in Figure 3.1 serve only to better illustrate the example. In

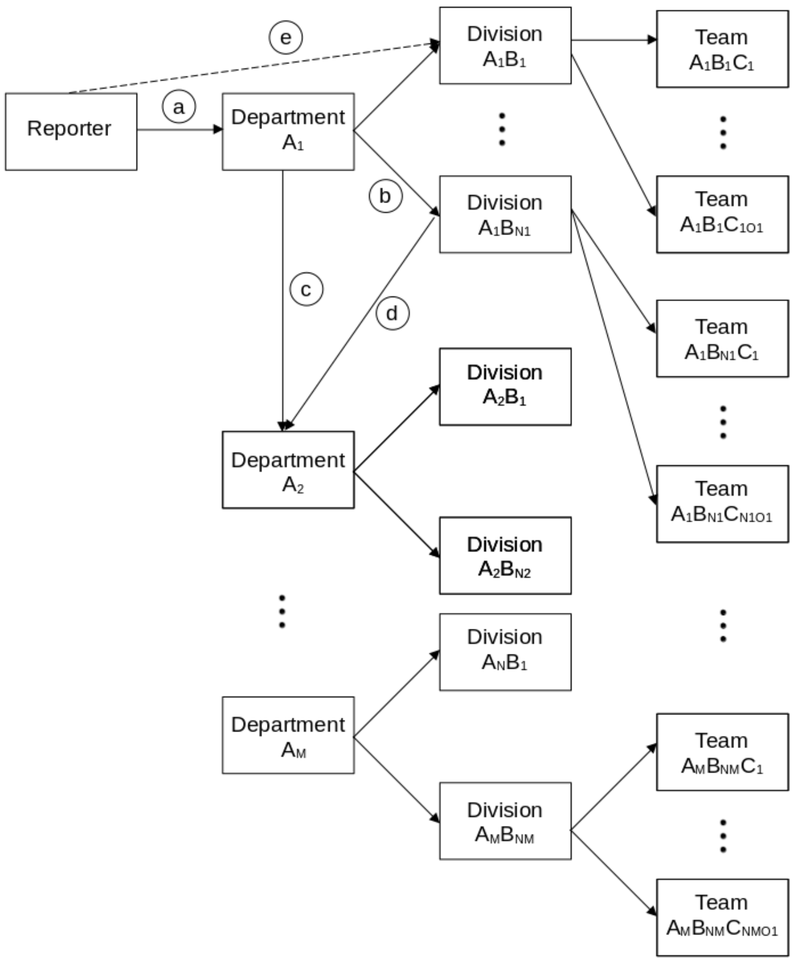


Fig. 3.1 Flow chart of process of transferring reports of bugs inside company, 3 layers shown

real use cases testers or customer support engineers decide which department is most suitable for resolving issue ⓐ, while reporting a software defect or anomaly. Next, a report is being assigned to one of the divisions inside the current department ⓑ or transferred to another one ⓒ. The problem of assigning a software bug report to correct group in that context may be interpreted as:

- assigning to department ⓐ or ⓒ,
- assigning to division in context of department ⓑ,
- assigning to division directly ⓔ.

3.2 Related Works

There is a plethora of ways to classify issues, i.e., classifying severity [55] or assigning the issue to a group which should handle particular case. As there may be numerous bug reports, not all of them are handled simultaneously. Among others based on classification of severity decisions are made as to whether the bug will be fixed now, later or never. In [70] an approach to assign issue to specified components is presented. In above-mentioned work authors predicts if reassignment of created bug report will occur. For that purpose they are using data which come from major projects Eclipse and Mozilla. In [19], bug report assignment is done directly to developers. In the scope was to build time oriented expert model which assigns more priority to developer who had worked on the similar bugs in the past. There are created activity profiles of people who deliver corrections with usage of factor for normalizing which uses the time of last usage of term by developer. [38] addresses bug report assignment to departments. It uses a specific time dependencies for creating train and test sets. Also one of the scope of that work was to investigate the impact of different way of preprocessing and vectorization on bug assignment accuracy. [112] considers bug report assignments to development teams. Moreover, the approach presented in [112], uses only selected cases to automate bug triaging. Selection is made based on confidence of the prediction. A threshold (cutoff) for the confidence is used as there exists a trade-off between accuracy and the number of predictions. [41] presents dual-output deep neural network which simultaneously predicts developer and team. Authors of that work also indicate the fact that this approach is robust against organizational changes as relations between teams and developers may change. In different kind of applications like disease detection and classification, a hierarchical concept of combination of machine learning models is used in [18]. There are used two layers. The purpose of the first of them is disease detection and seconds one is its classification. The results also suggest that the hierarchical approach can outperform the flat one, especially in case of small amount of data. A general concept of hierarchical classification algorithms is described in [28]. It presents among others different type of structures for hierarchical problems.

Although in [112] was introduced the possibility of transferring bug reports to selected organization parts based on the thresholds, there is lack of publication which applies specific context and additional possibilities which can be gain due to known hierarchical structure, like for instance combination of models related to different levels especially with usage of tuning of thresholds.

3.3 Research questions

- RQ.D.1: What are results of human predictions on department and inside department levels?
- RQ.D.2: What is the relation between results of machine learning predictions versus human predictions?
- RQ.D.3: What are the factors impacting human predictions?
- RQ.E.1: What are minimal requirements for solution applicability in software

development company?

- RQ.E.2: What are the scenarios of deployment of application?
- RQ.E.3: What are main advantages and disadvantages against humans?

3.4 Proposed solution

The novelty is that the cases incoming into department are being transferred into divisions (operation (b) in Figure 3.1) only in specific conditions. In that case a novel combination of machine learning models may be used, where one of the models predicts which division the issue should be addressed to, whereas the second one predicts whether the current department is the proper one. As a result, it is a transfer to proper division only in case if both of used models exceeded respective threshold. By the threshold we understand the cutoff confidence score based on the output of machine learning model. The novel features can be expressed as follows:

- assigning only selected cases of all cases incoming into department (or created inside) to specific divisions based on confidence level of prediction or state of issue,
- creating a model or other decision system based on at least two classification models, where at least one of them predicts division and at least one of them predicts department.

That approach is general and it is not limited to one way of creating a model. In a specific implementation in the company, predictions come from models prepared in a way similar to the approach presented in [38] . Fields used from bug report are title, description, product and release. The preprocessing phase uses methods for cleaning the text like removing chosen special characters, changing to lower letters, removing, stopwords and part of content related to company template. The training set is built from data related to relevant cases from last 365 days up to date of creation of model. For each case the result of prediction was collected at the time when formally correct bug report appeared for the first time at A_1. Different models are used for predictions at different levels of organization. Department is being predicted with the use of logistic regression classifier; division with use of support vector classifier with linear kernel. This production setup is being updated daily to get the newest available data for training as fast as possible.

3.5 Results

The doctoral student conducted research on the possibility of automatic assignment of bug reports in selected cases. The results of the studies show what the effect on historical data would be if bug reports were sent to inside department from interface group A_1. The predictions come from are cases which passed formal check of correctness of report filling at the time of use of machine learning model to make prediction and contain valid log content. Table 3.1 shows the number of cases above a certain threshold of a model which would be sent predicting department and respective precision. Similar research was conducted for model which predicts divisions inside the department. Table 3.2 indicates the number of cases which would be transferred from group A_1 to A_1B_1 and its precision. The results with combination of those two models are placed in Table 3.3. Based on above data, the decision about implementation pilot solution with thresholds 0.6 for department, 0.3 for division was made. Within that solution problem reports which meet the above-mentioned requirements were transferred. For those cases which did not meet these conditions were only placed information about suggested transfer possibilities. Selected results are presented in Tables 3.4 to 3.6. Presented data do not show cases which were created at early stages of software development and discovered in later phases or even in customer use. The following notations are used in Table 3.6:

- Human only - percentage of cases where human prediction was correct, but ML model prediction was incorrect;
- ML only - percentage of cases where ML model prediction was correct, but human model prediction was incorrect;
- ML & Human - percentage of cases where both ML and human model predictions were correct;
- Both incorrect - percentage of cases where neither ML nor human model predictions were correct.

.

Additionally, we can see the benefits like that for cases in date ranges from November 2021 to January 2022 where the decision about transfers was made 79% of them were resolved[1] (or fix was not required[2]) inside department A_1, but for cases where only decision about suggestion was made only 66% were resolved (or fix was not required) inside.

[1]Resolved - resolved; not including internal department cases; ended inside department

[2]Fix not required - fix not required; not including internal department cases; ended inside department including inflow group A_1

Table 3.1 General flow of issues in organization

Threshold set	Cases predicted as A_1	Precision of A_1 in the context of cases above threshold
0.2	266	58
0.3	244	61
0.4	183	64
0.5	130	68
0.6	71	67
0.7	48	71

Table 3.2 General flow of issues in organization

Threshold set	Cases predicted as A_1B_1	Percent of cases A_1B_1 among accepted ended on RD_RF_	Percent of cases A_1B_1 among accepted
0.2	206	39	20
0.3	148	40	23
0.4	95	42	23
0.5	53	38	23
0.6	28	39	25

3.6 Discussion on requirements for application of solution inside company

3.6.1 Minimal requirements

Although many people at glance think that such solutions have opportunities to be introduced only in case the predictions are better than human ones, this is not so simple as it is thought. In the case when machine learning predictions are better than it is rather obvious that is worth to make it application. Otherwise when it is worse, in some cases it may be also worth developing and applying such solutions. One of the reasons is that it may work as decision supporting system which does not make a binding decision, but only delivers suggestions which may be helpful for cases when a reporter has no idea how to address the problem, and sometimes may ignore suggestions when is sure where to address that or knows that the suggested target is wrong. Sometimes an issue with group overloading may occur, like for instance they currently handle too many bug fixes simultaneously, or have to deliver already committed new important features to product. Then, from the businesses perspective it may be reasonable to redirect cases to groups where it is less likely that the corrections will be delivered, but they may de-

Table 3.3 General flow of issues in organization

Threshold set for department A_1	Threshold set for A_1B_1	Cases predicted as A_1B_1	Precision of A_1B_1 in the context of cases above threshold and ended in A_1	Precision of A_1B_1 in the context of cases above threshold
0.3	0.3	88	36	25
0.3	0.4	60	38	26
0.3	0.5	40	38	25
0.3	0.6	26	41	27
0.4	0.3	65	34	25
0.4	0.4	44	34	25
0.4	0.5	31	34	26
0.4	0.6	21	33	24
0.5	0.3	49	40	29
0.5	0.4	34	40	29
0.5	0.5	21	43	33
0.5	0.6	13	44	30
0.6	0.3	29	57	41
0.6	0.4	20	64	45
0.6	0.5	11	75	55
0.6	0.6	7	80	57

liver detailed analysis or reject bug report as not valid. That effect may be gained due to tuning of mentioned in this work thresholds. Even if this change could lead to accuracy reduction, it may help to achieve business goals.

3.6.2 Human factors

There are usually many validation aspects when comparing machine learning and human predictions in software bug report assignment process. Chosen of them are presented in the following paragraph. One of the most important ones is that the reporters may use an already introduced decision supporting system. The different aspect is that in cases when multiple groups delivered correction people can choose which one will be the final main one and may want to boost human or machine learning result if they wish so. What is more, reporters sometimes ask before creating reports where reports should be sent before creating formally one. At that step, many developers might be involved or even the solution might be known before the actual report is officially processed. Sometimes developer teams ask for verification of some functionality and create a report directly against them. In those last two cases the final group is known even before creating bug report. What is more, not always the best accuracy is the aim

Table 3.4 Chosen results transferred cases based on ML model decision

Date range	type of resolution	Accuracy
November and December	Resolved	38 %
November and December	Fix not required	55 %
December	Resolved	50 %
December	Fix not required	80 %

Table 3.5 Chosen results of suggested cases based

Date range	type of resolution	Human accuracy	ML model accuracy
November and December	Resolved	54 %	38 %
November and December	Fix not required	65 %	35 %
December	Resolved	56 %	38 %
December	Fix not required	64 %	32 %

of introduction such solutions. Last, but not least, recently detailed instructions on how to address the most common types of bug reports and responsibilities of divisions inside department were made to improve human decision making.

3.6.3 Advantages and disadvantages of such solutions

The main disadvantage of such solutions, which are often pointed out during automatic transfers, is the lack of analysis that would provide information on why such a decision was made. There is already a paper [39] on the possibility of using explainable artificial intelligence to deal with that problem. The second issue that will help minimize these defects is the implemented solution, which conveniently displays to developers' key information about the content of the base station configuration state logs at the time of collecting logs, provided of course the logs have collected in the correct way.

3.7 Next steps which were made

Referring to the progress of implementation in industry, prepared an earlier pilot solution supporting the group A_1, dealing with the handling of applications within the department in which the doctoral student works, was gradually extended to handle more bug reports. The solution was to transfer selected bug reports from the group symbolically marked as A_1 to selected groups in the conditions specified by the machine learning model and if met the formal conditions for notification. This decision shall be taken automatically without the need for human verification. The model prepared was also used for transfers from department A_2 to the teams (A_1B_x) responsible for analysis in department A_1 as well as transfers to department A_2. One of the models that the doctoral student prepared is also used as one of the component models to solve suggesting to the reporter whether the report should be opened against department A_2. In addition,

Table 3.6 Distribution of specific types of results of suggested cases

Date range	Type of resolution	Human only	ML only	ML & Human	Both incorrect
Nov and Dec	Resolved	31 %	15 %	23 %	32 %
Nov and Dec	Fix not required	40 %	10 %	25 %	25 %
Dec	Resolved	34 %	16 %	22 %	28 %
Dec	Fix not required	40 %	8 %	24 %	28 %

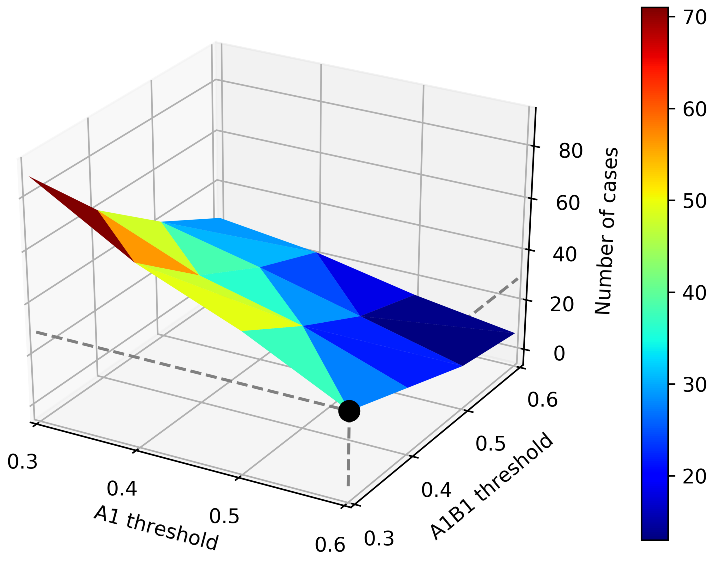


Fig. 3.2 Number of bug reports meet threshold conditions in function of given thresholds

it is also used as a component model for the automatic transfer solution applications between departments A_2 and A_3. There was decided to remove a group responsible for initial investigation inside that department A_1 in June this year, thus fully abandoning one layer of analysis. In connection with these changes, the doctoral student adapted the solution so that the submitted applications from department B were sent directly to groups A_1B_x . In addition, the system had to be adapted to indicate new groups after organizational changes, because on this layer the structure has also changed.

3.8 Summary

The paper discusses problems related to methods of assignment of reports, feature requests, supporting questions or similar issues to group of employees, developers, orga-

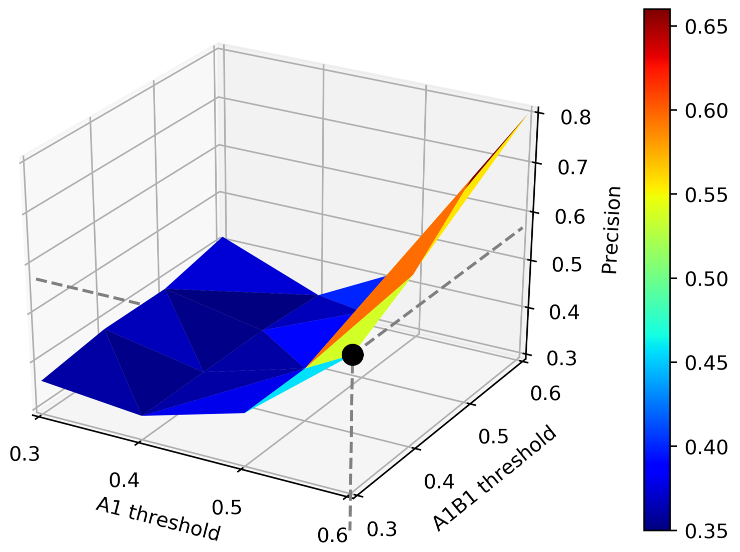


Fig. 3.3 Precision of predictions of bug reports meet threshold conditions in function of given thresholds excluding cases which ended outside of department A_1

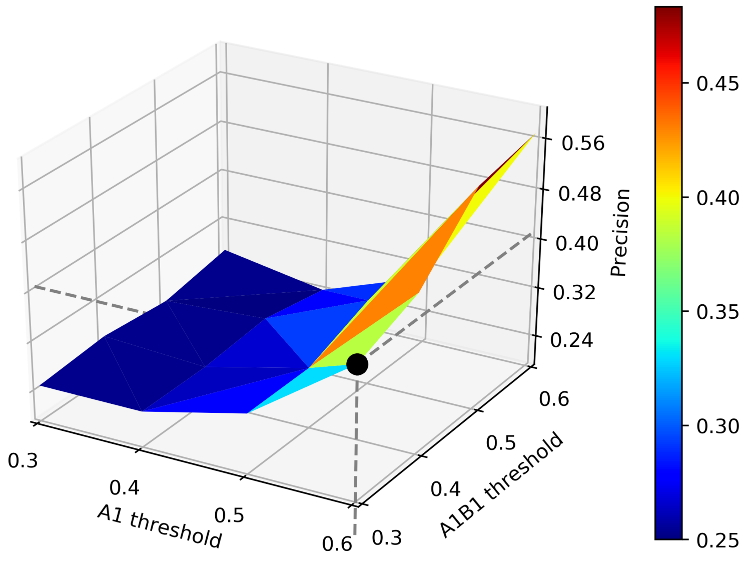


Fig. 3.4 Precision of predictions of bug reports meet threshold conditions in function of given thresholds including cases which ended outside of department A_1

nization unit, etc. The novelty introduced in this paper is related to the specific usage of the organizational structure in processes of handling (assigning) an issue. The paper shows possible scenarios of deployment of application supporting fault management with the use of solution based on machine learning. The study demonstrates factors impacting human predictions, main advantages, and disadvantages of automated solution against human. Comparison of results between human and model predictions at both department and inside department levels are presented. Minimal requirements for the company in case of application of machine learning supporting system in the company are also defined.

CHAPTER 4

Novel method of building train and test sets for evaluation of machine learning models related to software bugs assignment

Nowadays many tools are in use in processes related to handling bug reports, feature requests, supporting questions or similar related issues which should be handled during software development or maintenance. Part of them use machine learning techniques. In introduction is presented a review of fundamental methods used for evaluation of machine learning models. This paper points out weak points of currently used metrics for evaluation in specific context of the cases related to software development especially bug reports. The disadvantages of state of the art are related to disregarding time dependencies which are important to be applied for creating train and test sets as they may have impact on results. Extensive research of the art has been conducted and has not been found any article with the use of time dependencies for evaluation of machine learning models in the context of works related to software development applications like machine learning solutions to supporting bug tracking systems. This paper introduces a novel solution which is devoid of these drawbacks. Experimental research showed the effectiveness of the introduced method and significantly different results obtained compared to the state-of-the-art methods.

4.1 Background of the study

During the development of various types of systems, including software and those related to the hardware part, it is inevitable to make mistakes. In the event of noticing unexpected behavior of the system, testers or users create bug reports. Such a report may contain the contents of the log, screenshots, photos, reports from the spectrum analyzer, etc. Reporters should include information related to the discrepancy between the expected operation of the solution and the actual results obtained. This discrepancy may be the result of, e.g., a software malfunction, hardware failure, or environmental factors. Such a report must be assigned to a group of engineers for further analysis. This activity can be supported by machine learning solutions.

4.1.1 Problem statement

The paper discusses different methods of evaluation of results of machine learning predictions related to reports of bugs, feature requests, supporting questions or similar related issues which should be handled during software development or maintenance. For instance, it may be evaluation of machine learning predictions of bug reports assignments. There is plethora of ways to classify issue or bug report as for instance classify severity in article [55] or assign it to group in which should handle cases in papers [19, 70]. The problem raised in the article concerns about use for evaluation in these specific applications time dependencies with usage of for instance: date of issue creation, date of solving issue or assigned states with corresponding dates of changes. Time from creation bug report to solving the case may, contrary to appearance, take a long time. For example, it may take a day or two to resolve a problem, but on the other hand, some cases are resolved after more than a year. Transition states might be also used in cases like an issue that has been marked as solved and later reverted from that state due to finding that delivered fix had been working only partially.

4.1.2 Organization of the chapter

The chapter is organized as follows.

Section Introduction contains information about background of the study, problem statement, related works, motivation including research gap. At the end of this part, the work contribution and its significance are shortly summarized. Next section, Methods, begins with ways of presenting machine learning results with the use of confusion matrix and description of state-of-the-art methods for building train and test sets in the context of software bug reports assignment. The section also discusses novelty in building train and test sets in the context of software bug reports assignment. The chapter ends with sections Results and Discussion and Conclusion.

4.1.3 Related works

There are plenty of publications related to handling of reports of bugs, feature requests, supporting questions or similar related issues which should be handled during software development or maintenance. None of those publications consider the influence of time dependencies related to date of reporting and solving software bug reports on evaluation methods which are used in them. However, in these publications, state-of-the-art methods that are not suitable for evaluation of machine learning models related to software bug reports have been used to evaluate machine learning tasks. Currently different approaches are being used, for instance precision used in this work is about a bug mining tool to identify and analyze security bugs using Naive Bayes and TF-IDF [26]. Combination of metrics like accuracy, precision and recall are applied in analyses with the aim to detect bug report duplication [117]. As there were no details about ways of data splitting into train and test sets, there were probably applied default assumption about random split. More advanced train and test sets creation methods, e.g., cross-validation, are also applied in problems related to software bug report assignment. As another example, there is a need to predict whether the first assignment of bug report is likely to be reassigned in the future [70]. In the second example [14], Latent Semantic Indexing for reduction of the dimensionality and Support Vector Machine for triaging bug reports is applied. Time based activity profiling of developers for creation of time oriented expertise model was investigated in paper [19] where top-k accuracy metric was used. Metric which takes into consideration a couple of best predictions was also used in work [94] which utilizes data from two types of inputs. Natural language description and discrete features separately. On non-textual inputs Principal Component Analysis is applied. For text data is utilized Entrophy-based feature selection. Bidirectional Long Short-Term Memory Network was considered to automate the bug triaging process in publication [86]. In material [38] standard methods for evaluation on train, test set are applied, but separated based on time dependencies in that way that reported earlier were used for train set and reported later used for test set. The problem of bug report duplicates is mentioned in works [82, 83]. The first is about possibilities of reducing redundant bug records, whereas the second about risk estimation among others considers predicting bug fix time. Publication [126] presents the statement that even after application of solution for just-in-time retrieval solution to avoid duplicates being created, there is still over 10% of bug duplicates in described Mozilla Projects. Authors of paper [69] even divides duplicated bug reports into two different categories. The first is related to reported bugs duplicates the master report of the same issue while the master problem was not yet resolved. The second category when during the report of issue master issue was already solved. Other works in the field of technical information technology and telecommunications where time dependencies are important are articles related to Quality of Service where topics like and latency re-transmission of data packets [17], latency [109] are considered.

4.1.4 Motivation and research gap

The research gap is strictly related to impact of time dependencies related to software bug creation and resolution dates with the use of machine learning techniques. Current methodologies do not employ these time dependencies. They are significant due to the fact that, in general, the problems solved in each department are expected to be similar to some extent, but we must bear in mind that the characteristics of the reported faults by software users change over time. The problem related to time dependencies is considered due to the fact that during software development, its behavior, the flaws it possesses, or its characteristics change. For instance, introduction into developed application new functionalities which are expected by the customers is accomplished by modification of existing source code. Therefore, in such cases new error numbers, configuration parameters, patterns of messages, alerts etc. may be introduced. Those parts when creating machine learning models lead to creating new features in representation of data like feature vector shown in Equation 4.1. For instance, related to new types of configuration parameters may be then introduced as new terms in term frequency representation. In real use applications from creating bug reports to solving ones take time. The models for production use are trained only with the use of resolved cases. Therefore, data representations used for creation of model for predictions at the beginning of introduction of each new functionality will miss at the time of creation of the first bug reports specific features related to them in vector representation like in Equation 4.1.

$$X = [x_1\ x_2\ x_3\ \ldots\ x_N] \tag{4.1}$$

The new features will not be introduced into representation until the first case is resolved and used for training of model for production purposes. Before that time, some similar cases may be reported, in real case application all of these will be predicted with the use of model being trained without those described examples on data which were currently possessed and labelled. Common approaches in ML applications utilize randomized ways of creating train and test sets. It may lead to a situation in which different samples referring to the same or similar case which were reported nearby will be present in both train and test set what is not possible in real case applications due to the above-mentioned restrictions. Therefore, the results of evaluation where those restrictions are disregarded may be significantly different. Those approaches should not be used for evaluation in those applications if the aim is to get results similar to that which we can obtain in production of solution. The main advance of the proposed method in paper being verified is that it reflects possible real-world scenarios.

Two research questions were explored:

- RQ.C.1: Are the standard machine learning methods for evaluation appropriate to evaluate problems related to bug handling?
- RQ.C.2: If not then what experimental protocol should be introduced?

4.1.5 Main contributions of research

The paper shows that state-of-the-art methods are not appropriate for evaluation of machine learning models in the context of cases related to software bug reports. Current solutions disregard time dependencies like creation and resolving date of issues related to software bug reports. What is not appropriate especially in case of results of predictions of software bug report assignment if the aim is to estimate what kind of results are possible to obtain in real production use. The outcomes of work are results of scientific research related to introduced in this paper original and innovative solutions of the scientific problem of evaluation of machine learning models in the context of software bug reports assignment. Introduced in this paper methods related to including time dependencies into evaluation of machine learning models in the context of software bug reports assignment do not impact accuracy of production solutions itself, however thanks to them the results better reflect real use cases. The presented innovative original solution in the field of application of research results are significant for the economic sphere.

4.2 Methods

4.2.1 Building train and test sets

In case of classification problems in machine learning to train model is used a set of data called train set. To test model is used a set of data called test set. Especially for applications related to neural networks sometimes is also used third type of set called validation set. The purpose of using that set may be for instance to check the state of neural network every epoch during the training phase and decide whether the training of the network meets condition for early stopping. In that situation the state of network is saved and used for further evaluation on separate test set, which has never been used during the test or evaluation phase of model.

4.2.1.1 Standard train test splitting

During common creation of train and test set for evaluation usually dataset is split into train and test set randomly.

4.2.1.2 Stratified train test splitting

The stratified version for division into multiple sets uses the information about classes and tries to keep the radio between the classes in sets as much as possible like each other.

4.2.1.3 No shuffle

In implementations like in library [99], there is an option to use a split of the dataset for train and test without shuffling. That might be applied in an application where the order of samples matters.

4.2.1.4 k-Fold Cross-Validation

Cross-validation is a procedure used to evaluate machine learning models which uses many splits on data. Generally, data is divided into k sets usually called folds. For each split one set is chosen as validation set, and the rest of data is used for training model in given split. After getting results for each split the results are summarized. Example of visualization of splits is shown in Figure 4.1 [29].

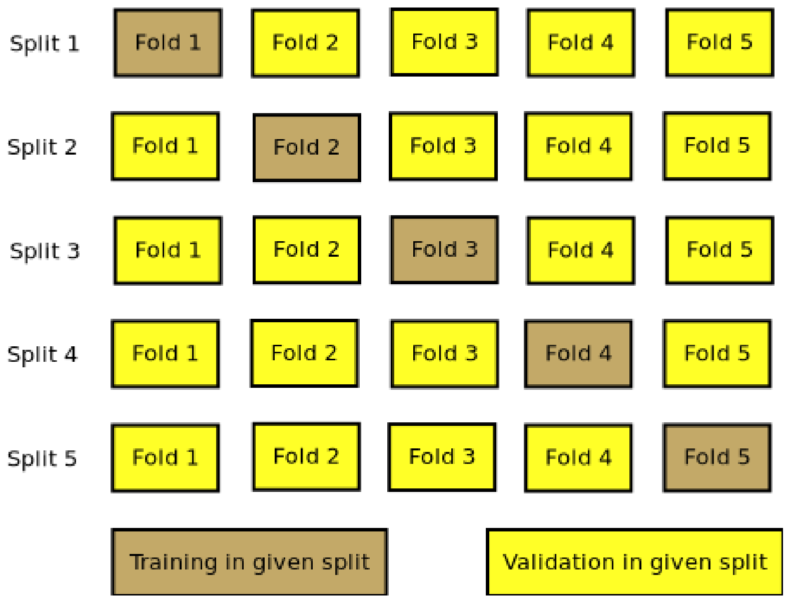


Fig. 4.1 Cross-Validation.

4.2.1.5 Leave One Out Cross-Validation

This procedure might be useful especially for small datasets. In that case the number of folds equals the number of samples in the dataset. The model is tested on single sample while trained using the rest of available data [111].

4.2.2 Novelty in building train and tests sets in the context of software bug reports assignment

The novelty is about specific splitting of train and test sets for purpose of evaluation to be more adequate and like results which we could expect from working production setup. While preparing model for production mode usually only solved cases with respective final correct labels are used for creating train set. The advance is taking for evaluation cases separated with the use of time dependencies. An example of data dependency is shown in Figure 4.2. For sake of simplicity of presentation each case is identified with the id like A, B, C, $\ldots$ It is only a identifier and it is not a label/class in the context of Machine Learning task. Figure 4.2 shows an example diagram with issues named A, B, C, D, E, where A_R is point where case A was reported, A_S is a point where case A was solved. The proper label/class related to group(s) which were engaged in solving case used for training can be assigned after the case is solved. According to this example the split point is marked as t_1. For train data cases A, C were selected as were solved before t_1 and for test set selected cases reported after t_1 (D, F). However, in some applications software bug reports which were reported before t_1, but not solved at the time of prediction might utilize machine learning supporting solutions to point out the proper group. Sequence diagram has been presented in Figure 4.3. There are shown interactions in chronological order between objects called setOfIssues, machineLearningModel, reporter, and developer. The diagram clearly shows accessible data over time which can be utilized for model training. Dataset of issues is updated there by developer at the time of resolving issues when the labels are assigned, what triggers retraining of model which can be accessed by reporter. More complex solutions may utilize also the time needed for real use application to deliver new model for production Δt. For evaluation it might be also used with derived approaches using results from multiple splits by moving time division point or moving windows using this fact. After calculating results for multiple splits, they may be averaged or analyzed in another way for better presentation of results. These time dependencies are important for evaluation especially when taken into consideration duplicates and duration of solving issue related to bug report. In case of random split and other standard machine learning methods for creating train and test sets like these in source [99], these time dependencies are skipped. This fact may impact on results, because model used for production purpose cannot be trained with the recently reported or not yet resolved bug reports that situation is not the case in random split evaluation especially when there is a lot of bug report duplicates. We should expect that in general when applying these time dependencies, the results which would be obtained will be worse than with standard methods which do not meet those requirements like Cross-Validation. At the same time, we have to remember that the results which were obtained using methods which do not meet the requirements of time dependencies do not show results which are similar to that what might be obtained in production of such solutions.

In practical applications such models may be retrained daily or even more frequent to minimize the effect. Moreover, in such cases the window which is used for training might be fixed by the beginning date, time duration, number of samples in the window or even more complex to somehow adjust the distribution of classes inside training set.

4.2.2.1 Set of novel methods

$$\overset{t_S}{\underset{t_R}{S}} \ ; \ t_S - \text{date of solving} \ ; \ t_R - \text{date of reporting} \ ; \ S - \text{Software Bug Report} \quad (4.2)$$

$$\mathbb{X} = \left\{ \overset{t_S}{\underset{t_R}{S}} : t_S < t_D \right\} \ ; \ \mathbb{X} - \text{Train Set} \ ; \ t_D - \text{time of division} \quad (4.3)$$

$$\mathbb{Y} = \left\{ \overset{t_S}{\underset{t_R}{S}} : t_R > t_D \right\} \ ; \ \mathbb{Y} - \text{Test Set} \quad (4.4)$$

$$\mathbb{Z} = \left\{ \overset{t_S}{\underset{t_R}{S}} : t_R < t_D \wedge t_S > t_D \right\} \ ; \ \mathbb{Z} - \text{Spare Data} \quad (4.5)$$

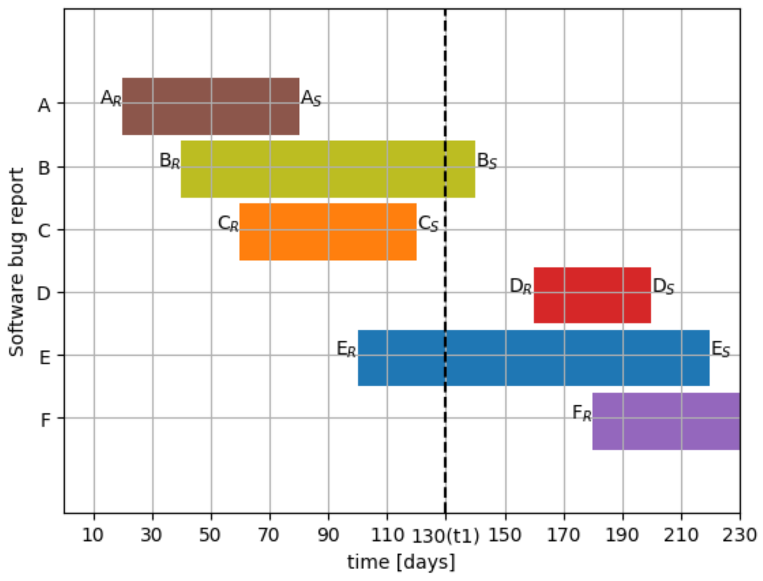


Fig. 4.2 Timeline of bugs (reported and solved if applicable).

Single division point:

In the novel approach metric for evaluation of machine learning models related to software bug reports assignment is being calculated with the use of single split point t_D used for building train and test sets for evaluation. Introduced symbols and general definitions of sets are presented in Equations (4.2) to (4.5). Division point t_D can be selected arbitrarily, e.g., $t_D = t_{p80}$, where t_{p80} is a date of reporting case in about 80th percentile of dates of creation. Let assume that the calculation of metric in single point will be symbolized with $\lambda_{metric}\left(t_D\right)$ for instance $\lambda_{acc}\left(t_{p80}\right)$ (See Equation (4.6)). The point t_1 does not have to be chosen randomly, it may be selected in a different ways depending on needs.

$$acc = \lambda_{acc}\left(t_{p80}\right) \tag{4.6}$$

Multiple division points:

There might be multiple solutions for averaging metric with the use of multiple division points, for every point or for instance with the use of moving window which can be defined both by time, or number of issues, or even some kind of stratification. In Equations (4.7) to (4.10) is defined example with the use of division for every reasonable point. However, in practice, the starting point should not be one of the first, given the chronological order.

$$\mathbb{T} = \left\{ t_R : \underset{S \in \{\mathbb{X},\mathbb{Y}\}}{\forall} \overset{t_S}{\underset{t_R}{S}} \right\} \cup \left\{ t_S : \underset{S \in \{\mathbb{X},\mathbb{Y}\}}{\forall} \overset{t_S}{\underset{t_R}{S}} \right\} \tag{4.7}$$

$$\mathbb{Q} = \mathbb{T} \setminus \left\{ min\left(\mathbb{T}\right), max\left(\mathbb{T}\right) \right\} \tag{4.8}$$

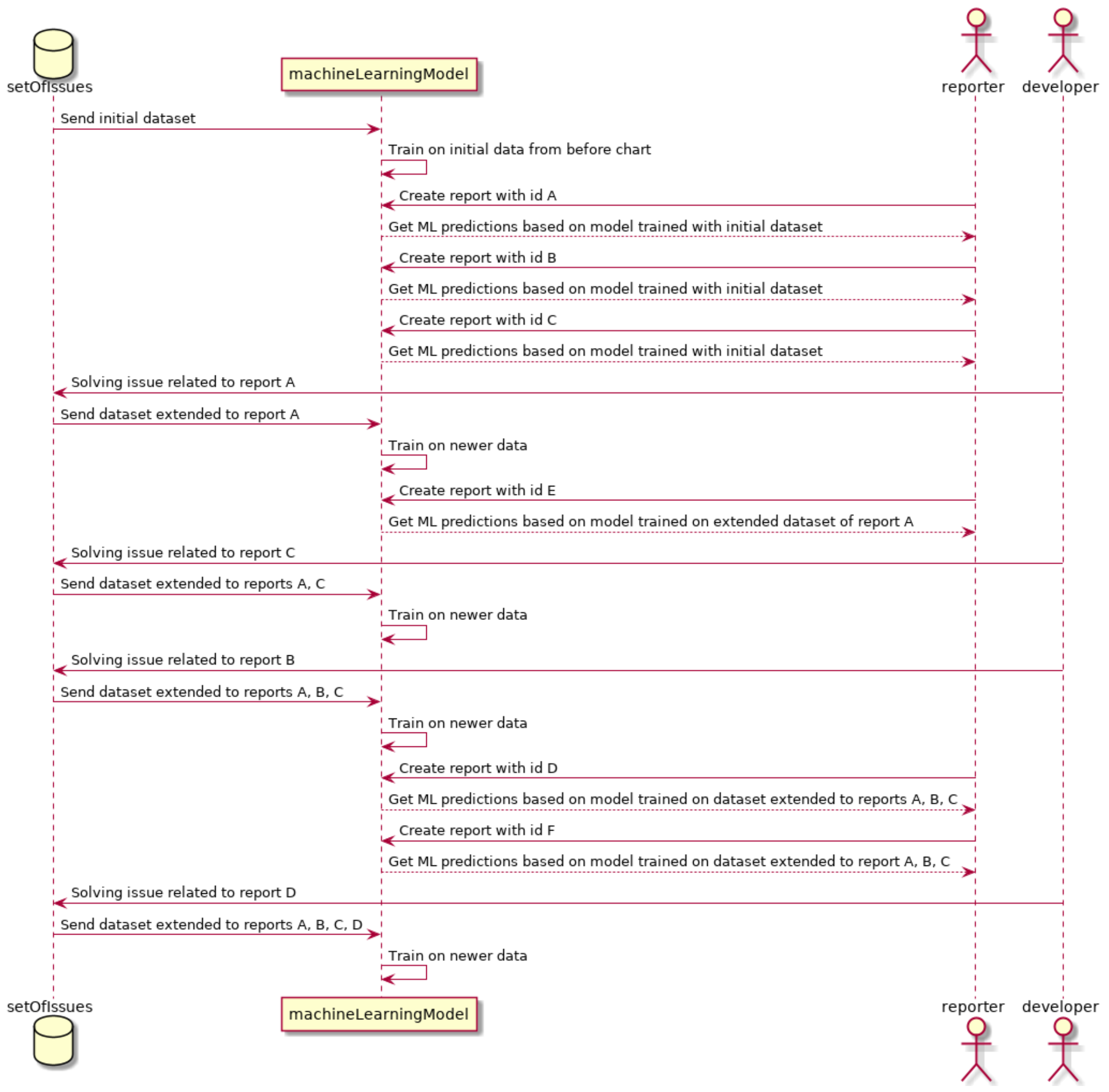

setOfIssues
machineLearningModel
reporter
developer
Send initial dataset
Train on initial data from before chart
Create report with id A
Get ML predictions based on model trained with initial dataset
Create report with id B
Get ML predictions based on model trained with initial dataset
Create report with id C
Get ML predictions based on model trained with initial dataset
Solving issue related to report A
Send dataset extended to report A
Train on newer data
Create report with id E
Get ML predictions based on model trained on extended dataset of report A
Solving issue related to report C
Send dataset extended to reports A, C
Train on newer data
Solving issue related to report B
Send dataset extended to reports A, B, C
Train on newer data
Create report with id D
Get ML predictions based on model trained on dataset extended to reports A, B, C
Create report with id F
Get ML predictions based on model trained on dataset extended to report A, B, C
Solving issue related to report D
Send dataset extended to reports A, B, C, D
Train on newer data
setOfIssues
machineLearningModel
reporter
developer

Fig. 4.3 Sequence diagram presenting time dependencies of real use case in the context of solution related to software bugs assignment systems.

$$metric = \frac{\sum_{t \in \mathbb{Q}} \lambda_{metric}(t)}{card(\mathbb{Q})} \tag{4.9}$$

$$acc = \frac{\sum_{t \in \mathbb{Q}} \lambda_{acc}(t)}{card(\mathbb{Q})} \tag{4.10}$$

4.2.3 Machine learning metrics and ways of presenting results

4.2.3.1 Confusion matrix

Confusion matrix is used for presenting information related to results of machine learning predictions. In columns are presented predicted classes, in rows actual classes. That way of orientation of matrix is used in many sources[22, 24, 47, 57, 100, 115, 123, 129], however different sources [21, 49, 64] use another. This means that adding the right

		Predicted class			
		Department K	Department L	Department M	Department N
Actual class	Department K	$\frac{X_{KK}}{\sum X}$	$\frac{X_{KL}}{\sum X}$	$\frac{X_{KM}}{\sum X}$	$\frac{X_{KN}}{\sum X}$
	Department L	$\frac{X_{LK}}{\sum X}$	$\frac{X_{LL}}{\sum X}$	$\frac{X_{LM}}{\sum X}$	$\frac{X_{LN}}{\sum X}$
	Department M	$\frac{X_{MK}}{\sum X}$	$\frac{X_{ML}}{\sum X}$	$\frac{X_{MM}}{\sum X}$	$\frac{X_{MN}}{\sum X}$
	Department N	$\frac{X_{NK}}{\sum X}$	$\frac{X_{NL}}{\sum X}$	$\frac{X_{NM}}{\sum X}$	$\frac{X_{NN}}{\sum X}$

Table 4.1 Normalized confusion matrix for multiclass problems.

headings in this kind of presentation is very important to avoid misunderstanding. Additionally normalized way of confusion matrix is presented in Table 4.1. By normalization means the division of each element in the matrix by the sum of the samples ($\sum X$).

4.2.4 Description and experimental protocol

For below described experiments were performed calculations with four different methods of evaluation:

- split for train and test set randomly with shuffle of data (20% for test data);
- Cross-validation (5 folds);
- split for train and test set with the use of only date of reporting (8 months for train set, following 2 months for test set);
- split for train and test set with the use of novelty so both data of reporting and solving was taken into consideration (8 months for train, following 2 months for test).

Each experiment contains data from the range of 10 months. Please note that in the last of evaluation methods cases reported within the first 8 months and resolved later cannot be taken into consideration and were removed. The task performed during the experiments is to assign the report of bug to proper department responsible for investigation or solving issue. For performing that research only cases where fixes have been delivered have been taken into consideration. All calculations have been performed with the same way of preprocessing, with the same parameters to build TF-IDF representation. As a finial algorithm to assign department was used Logistic Regression. For each setting described above 10 series of calculations were performed with the move of dataset by one month between series.

4.3 Results and Discussion

Table 4.3 contains the results with the random split with shuffling. The measures which were presented are accuracy, weighted precision and weighted recall. The weight is related to the number of samples. Table 4.4 presents accuracy in case of Cross-Validation. Accuracy in the case of splitting data for train and test set with the use of time dependencies is shown in Tables 4.5 and 4.6. First of them with the only use of date of reporting of report, second with use of novelty for date of solving. Although the results of evaluation based on time split by creation dates includes dependencies relating to date of creation, they disregard the time of resolving the issues, therefore they do not obey the laws of physics. For each of ways of evaluation for the first series results are presented in normalized confusion matrices (Figures 4.5 to 4.8). From the results we can clearly notice that results with the use of novelty are significantly different than the rest of results which have been obtained. Comparison of accuracy has been also shown in the chart (Figure 4.4) and Table 4.2 for sake of transparency. For all series the results gathered with the use of novel method includes time dependencies between dates of creation and resolution of software bug report prediction accuracy is lower by at least fifteen percentage points by methods disregarding them. That novel method of building train and test sets for evaluation of machine learning models is the only one from those taken to comparison which meets the real use conditions. Mentioned dependencies are related to dates of creation and resolving the case. The model for evaluation should be trained only with cases which have labels assigned (in that case were resolved), before date of possible real usage in production (in that case date of creation software bug report). Noticing this fact and knowing that this method better reflects the production conditions of the applications of these methods, the thesis is put forward that it is better to use the method related to time dependencies and introduced novelty, if the aim is to reflect the results that can be achieved in real use.

4.4 Conclusion

The paper summarizes different methods of evaluation of machine learning models in the context of problems related to software bugs. Commonly used machine learning evaluation methods like random split, Cross-Validation and even standard splitting based on time like for instance based on date of creation of problem reports does not

Series	Random split	Cross-Validation	Time split by creation date	Time split with the usage of a novel time dependencies
s1	0.86	0.84	0.86	0.64
s2	0.85	0.83	0.85	0.63
s3	0.85	0.83	0.85	0.64
s4	0.87	0.85	0.87	0.64
s5	0.86	0.84	0.86	0.66
s6	0.86	0.85	0.86	0.69
s7	0.88	0.86	0.88	0.67
s8	0.87	0.85	0.87	0.66
s9	0.86	0.85	0.86	0.67
s10	0.86	0.85	0.86	0.66

Table 4.2 Comparison of accuracy.

Table 4.3 Detailed results of random split.

No.	Accuracy	Precision	Recall
1	0.86	0.87	0.86
2	0.85	0.86	0.85
3	0.85	0.86	0.85
4	0.87	0.88	0.87
5	0.86	0.87	0.86
6	0.86	0.87	0.86
7	0.88	0.89	0.88
8	0.87	0.87	0.87
9	0.86	0.87	0.86
10	0.86	0.88	0.86

include the time of solving issue what may have serious impact on results. In the paper was introduced a proposition to create train and test sets built based on time dependencies to create test set with bug report created not earlier than the latest date of solving of bug report from train set. The main advantage is that the results come from predictions in simulations which better reflect real use. Please note that although the results with the use of novelty may be significantly worse as they are in that case, the other ones are not reasonable due to breaking time requirements and should not been applied for such cases. Experimental results which were conducted in that work clearly show the difference between evaluation with the use of novelty and standard methods for general classification problems. Authors claim that the rest of the methods which do not meet time dependencies are not appropriate for evaluation problems related to software bugs reports as they do not respect real time dependencies.

Table 4.4 Detailed results of Cross-Validation.

No.	Accuracy	Precision	Recall
1	0.84	0.84	0.84
2	0.83	0.84	0.83
3	0.83	0.84	0.83
4	0.85	0.85	0.85
5	0.84	0.84	0.84
6	0.85	0.85	0.85
7	0.86	0.87	0.86
8	0.85	0.85	0.85
9	0.85	0.86	0.88
10	0.85	0.86	0.85

Table 4.5 Detailed results of split with use of the novelty.

No.	Accuracy	Precision	Recall
1	0.86	0.87	0.86
2	0.85	0.86	0.85
3	0.85	0.86	0.85
4	0.87	0.88	0.87
5	0.86	0.87	0.86
6	0.86	0.87	0.86
7	0.88	0.89	0.88
8	0.87	0.87	0.87
9	0.86	0.87	0.86
10	0.86	0.88	0.86

Table 4.6 Detailed results of split with use of the novelty.

No.	Accuracy	Precision	Recall
1	0.64	0.64	0.64
2	0.63	0.64	0.63
3	0.64	0.65	0.64
4	0.64	0.65	0.64
5	0.66	0.67	0.66
6	0.69	0.69	0.69
7	0.67	0.68	0.67
8	0.66	0.66	0.66
9	0.67	0.68	0.67
10	0.66	0.67	0.66

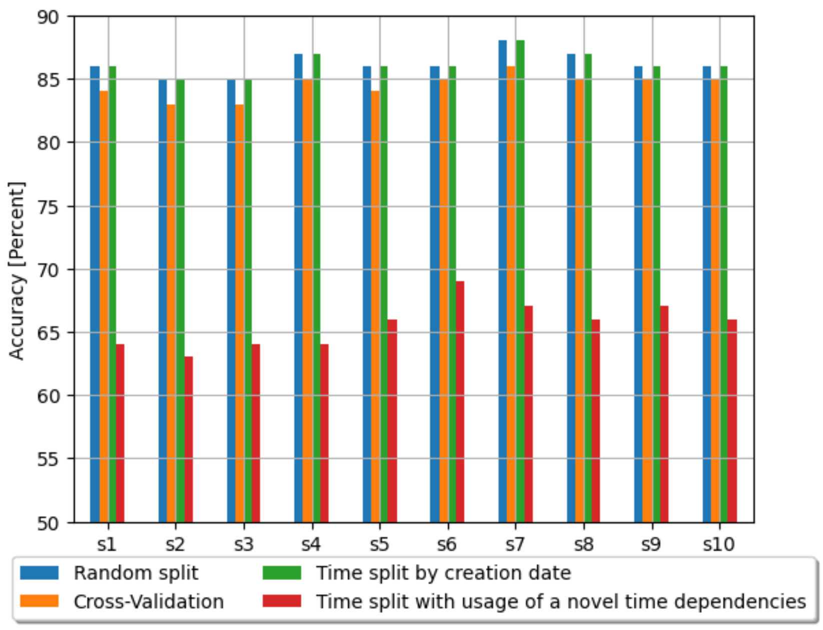


Fig. 4.4 Comparison of accuracy.

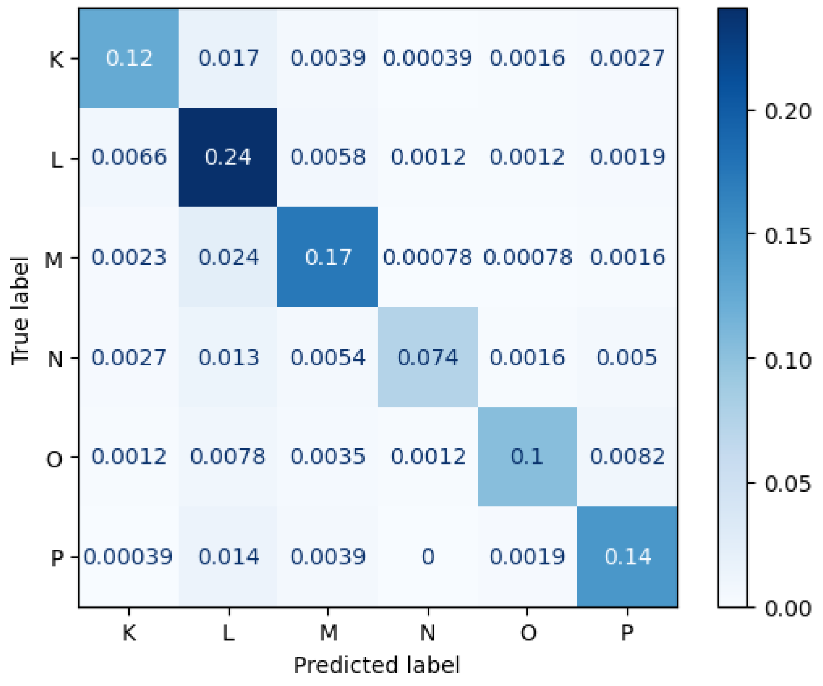


Fig. 4.5 Normalized confusion matrix random split.

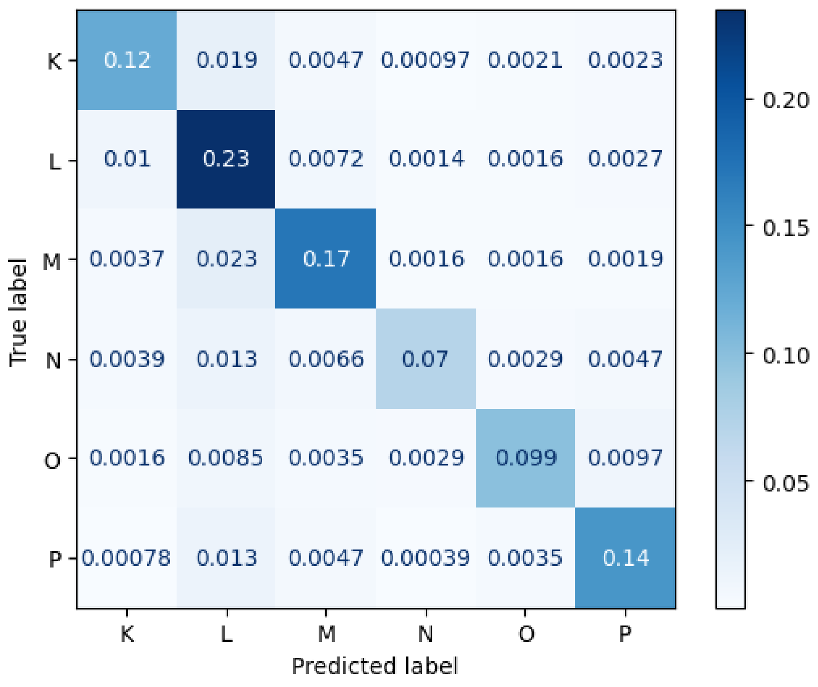


Fig. 4.6 Normalized confusion matrix Cross-Validation.

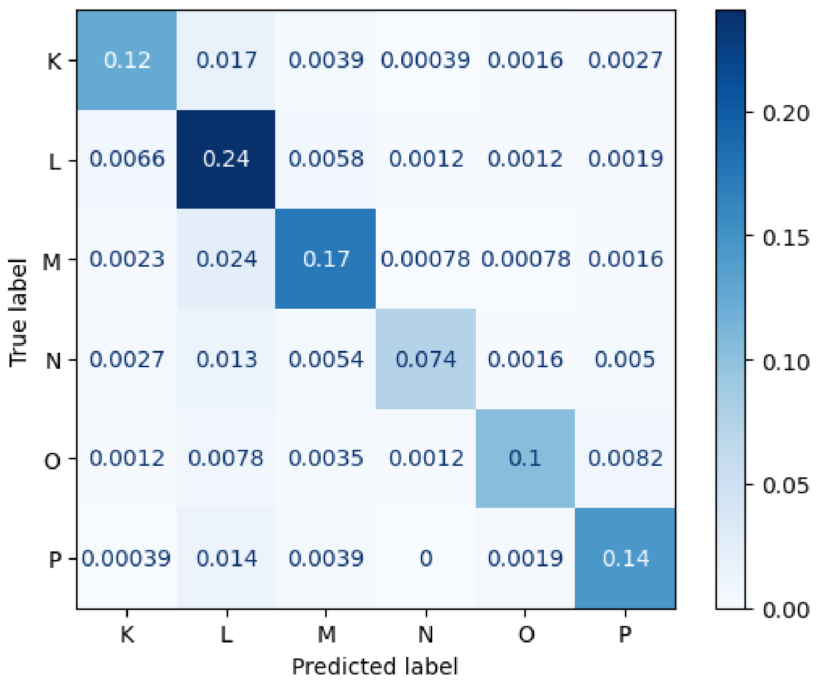


Fig. 4.7 Normalized confusion matrix by creation date.

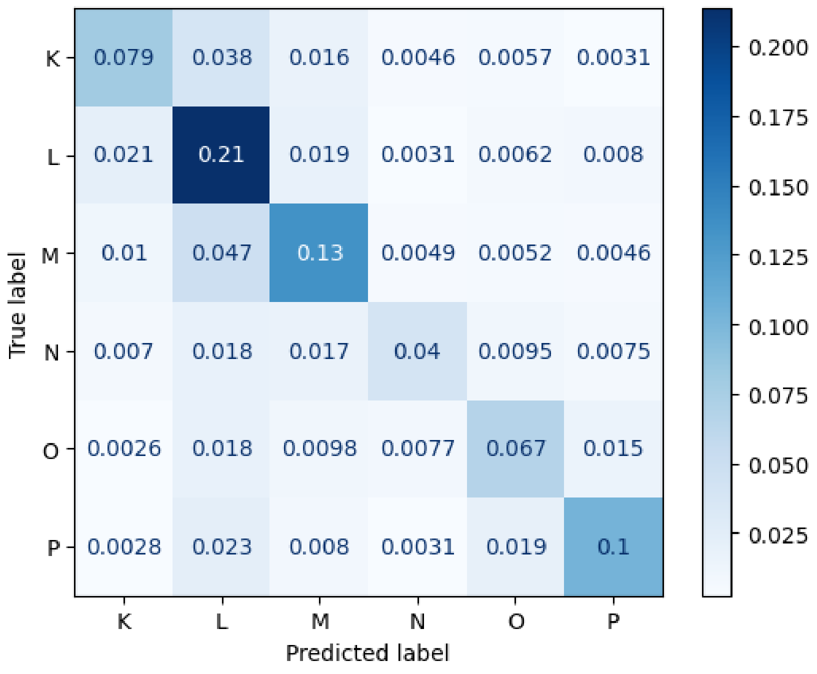


Fig. 4.8 Normalized confusion matrix with the use of novelty.

CHAPTER 5

Potential application of XAI

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.

5.1 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 [41]. In contrast to black-box solutions a proposition of application of one based on expert rules is shown in paper [19]. 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 [26]. Both methods used allow solutions to be explainable, but this circumstance was not used. The main aim was to analyze possibilities of applications of the explainable artificial intelligence (XAI) in specific case 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 [23]. 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 [120].

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 bugs. 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 [91] in cases where proposed patches are provided without explanations they are usually ignored. In that paper was a statement that those kind of systems which supports 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 [65]. There is a description of works related to predicting 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 [16] are shown results of application of model agnostic explanation methods like LIME and iBreak on bug prediction models. Paper [55] is about predicting the bug severity level and whether is not strictly related to XAI methods. At one of steps is used algorithm based on dictionary of critical terms related to appropriate severity level. That information might be useful to support 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 [32]. 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

Table 5.1 Distribution of security related and not security related issues (internal company data)

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

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 [84]. For extracting rules with a decision tree from black-box model as example may be an article [31]. The research questions which are being answered in the work are presented below:

- RQ.F.1: What is accuracy when comprising standard and easily explainable algorithms?
- RQ.F.2: What benefits that might gives?
- RQ.F.3: Does information provided by explainable models seems to be consistent?

5.2 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 quantity of samples could be provided, but information about distribution of data is shown in Table 5.1. Another example of article using data comes also from NOKIA is publication [38].

2. Mozilla Defect Dataset:

   Table 5.2 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 %

   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 5.2. Samples with multiple labels were removed. Generally, publicly available bug reports from Mozilla projects are accessible, among others, in repository [71] or can be gathered with script [80].

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, stop word then applying lemmatization. In the next step was applied vectorization 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 were performed calculations with usage of different algorithms. Results of selected standard algorithms used for XAI applications were compared against the 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.

5.3 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 5.5, 5.6, 5.7, 5.8. Headings used in Tables are:

- kNN - k - Nearest Neighbors;
- LR - Logistic Regression;
- NB - Naive Bayes;
- SVC - Support Vector Classifier;
- XGBoost - eXtreme Gradient Boosting;
- 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.

As is shown in table 5.3 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 5.4). It is noticeable in both of cases there are at least some of the same features are common for most important classifiers. This is also confirmed in Figures 5.1 and 5.3. Diagrams

Table 5.3 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

Table 5.4 Comparison most important features related to type od 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

(Figures 5.1 to 5.3) 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 5.3, 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.

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) for explainable were used for explainability com-

Table 5.5 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 5.6 Comparison of results related to label issue as security related or not (internal company data)

	5-5-10		5-5-15		10-10-15		3-3-15		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 5.7 Comparison of results related to type od 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 5.8 Comparison of results related to type od issue (Mozilla Defect Dataset)

	5-10-15		5-5-15		10-10-15		3-3-15		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	0.41	0.87	0.41	0.87	0.41	0.90	0.33	0.87

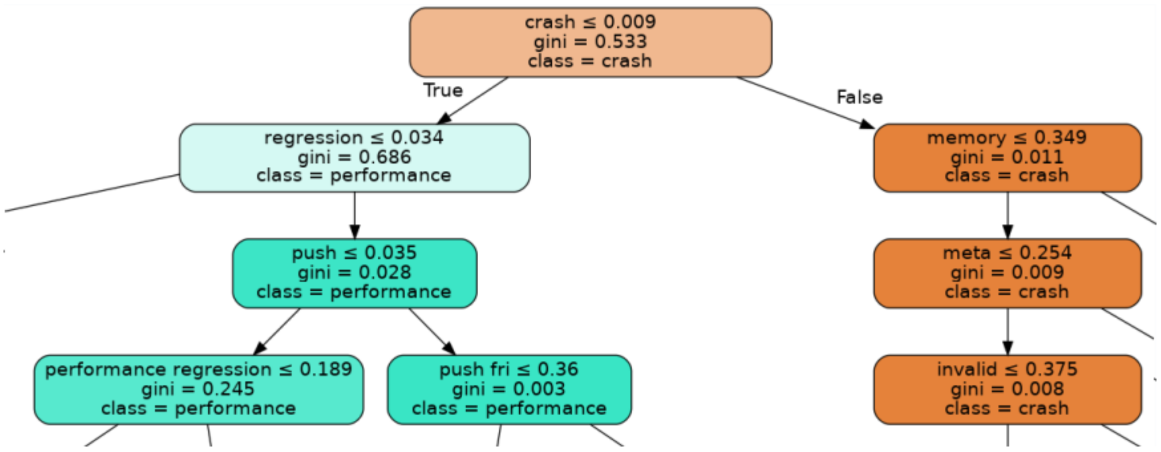


Fig. 5.1 Diagram of tree build on dataset from Mozilla to recognize types of issues.

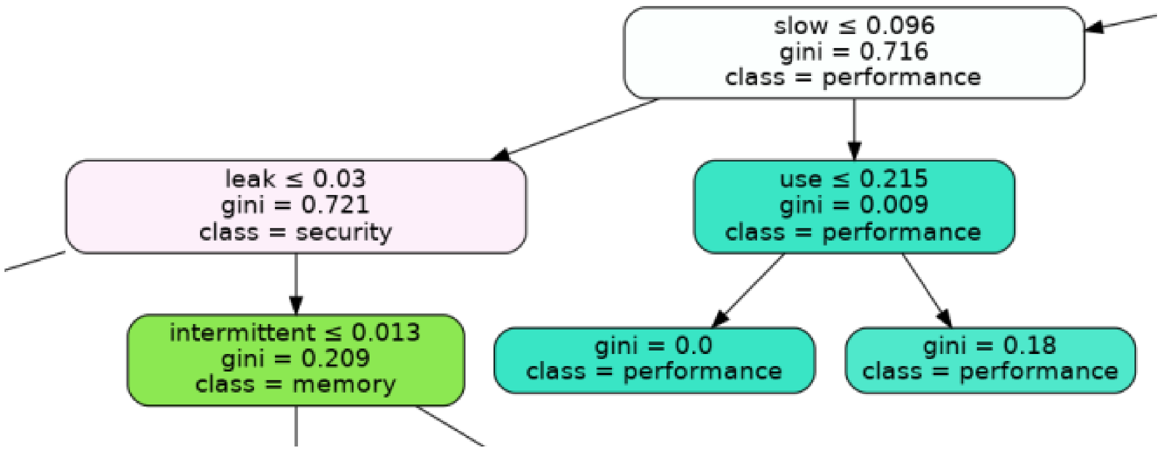


Fig. 5.2 Diagram of tree build on dataset from Mozilla to recognize types of issues.

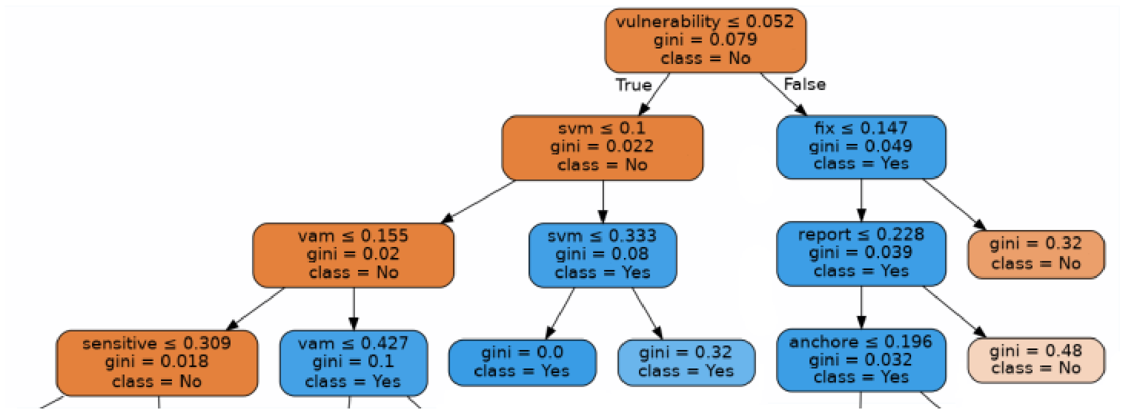


Fig. 5.3 Diagram of tree build on internal company dataset to recognize security related issues.

parison [23].

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 [105].

5.4 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 5.1 to 5.3) 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.

CHAPTER 6

Application of multimodal neural networks in solving problem of labeling bug reports

Handling bug reports is a crucial part of software development, especially for high reliability systems. Nowadays in processes which deal with these issues specific procedures using automated tools are introduced to enhance bug report assignment accuracy. Those tools might be enhanced with machine learning techniques. The purpose of this chapter is to examine the applicability of a multimodal neural network in a software bug assignment task based on diverse types of data, such as features extracted from logs, combination of natural language text fields such as title and description, and categorical information like product type. Computational experiments comparing results from multimodal neural network and reference methods like dense (fully connected) neural network is presented. Multiple series of experiments on different datasets have been prepared and performed with various configurations of neural network architectures as well different preprocessing settings for preparing data for respective input types. The charts in this paper show training and validation accuracy as a function of epoch of training and impact of introduced methods on minimization of overfitting. Results clearly showed that application of multimodal neural network brings benefits.

6.1 Introduction

In software development there are situations when software bugs or hardware defects should be handled. When a system malfunctions, user or tester prepares a report related to that issue. It contains the title of issue, description with the actual and expected behavior, steps to reproduce the issue if known. The case may rarely occur, may be not permanent and occur randomly or only in specific conditions, so it may be hard to reproduce the issue. Reporter also attaches the logs, sometimes they do not cover the issue (e.g., insufficient data collection duration time, inadequate data collection method like no debug flag added), and thus, additional log collection is required. The bug report should be assigned to a proper group in charge for further analysis of the case. It might be a team of developers or preinvestigators, department or division. It is usually specific to an organization or a company. The task of bug labeling differs due to the goal to achieve and might be different based on incomplete data and/or insufficient competences of people who make such decisions. These issues might be resolved by providing: label related to given department or division like in paper [37], or provide problem type label in article [39] as well as information whether is related to security or not. The datasets used in the study are not publicly available due to trade secrets of the company.

Currently according to the best knowledge of the authors there is no application of multimodal neural network in problems related to software bug assignment. This architecture has been considered because of its positive impact on practical applications with complex structure and heterogenous inputs. That problem is usually analyzed with fields like title and description. However, there are publications related to the use of some categorical fields like product type [38], [81]. There are also applications of neural networks in solving that task, but multimodal one has not been found. The paper [41] shows usage of two-output deep neural network architecture (Dual DNN) for predicting an assignment of a bug report to both developer and team. At the first step of the proposed method, the raw data is transformed with the use of latent semantic analysis, [72], reducing the input vector dimension to 1000. Following the input layer of neural network, there are two hidden ones. The last of them is connected to two output layers: teams and developers. There exists also one more connection from team layer to developer layer such that the team prediction information is used to predict developers. These two papers use summary and description from reported bugs to create a word document graph. Various neural network architectures have been applied to solving problems related to bug reports handling. The bug report assignment related problems are solved with the use of Long Short-Term Memory (LSTM), [58], in [116]. The bug report is preprocessed with the use of Word2Vec, [52], a natural language processing technique. Feature vectors are built using Senti4SD, [30], a sentiment analysis technique. That method is applied for bug prioritization. Convolutional Neural Network (CNN), [96], is utilized in [118] for automatic prioritization of bug reports. To create proper text representation some preprocessing techniques like stopwords removal, spelling correction, lemmatization are used. The work in this paper also employs Emotion analysis, [97], and word2vector [90]. The main machine learning model is based on a single Convolutional layer with multiple filters followed by pooling layer and fully connected layer with softmax activation as the output one. CNN has been also utilized in [130] for bug report assignment. The representation of issues was made with the two techniques which were used for comparison (Word2Vec and GloVe, [102]). The neural network for that purpose was created as follows: the input layer, convolution layer with multiple filters, max pooling layer, flattening and layer with softmax activation as output. A graph

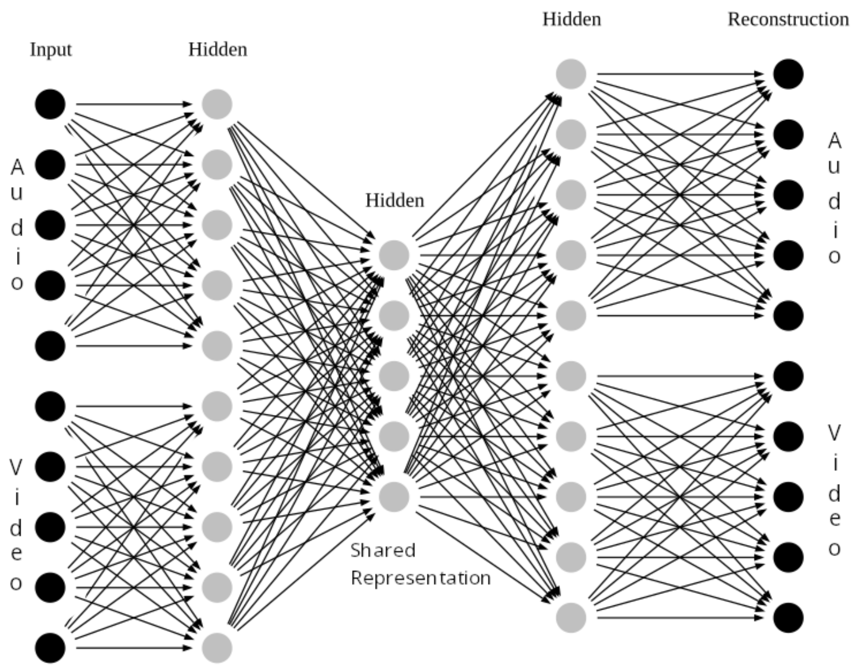


Fig. 6.1 Example of multimodal neural network.

representation method for the bug reports dataset is presented in [131] and [133]. This representation is further used for assigning bug reports. The paper [131] uses data related to natural language fields of bug reports like summary and description. The nodes represent documents (bug reports) and words. Edges are of two types: word-word co-occurrences and word-documents co-occurrences. Following heterogenous graph input, there are two layers of Graph Convolutional Network. For weighting the graph edges between bug document nodes and word nodes the term frequency-inverse document frequency (TF-IDF) is used, [107]. Weights of word-words edges are related to cosine similarity, [56]. [133] extends the paper [131] by adopting different similarity metrics and correlation metrics for weighting word–word edges.

The purpose of this chapter is to analyze applicability of multimodal neural network in task of software bug assignment based on data of diverse types like logs and combination of natural language text fields like title and description. An approach to train deep neural networks using data coming from multiple modalities was introduced in [95]. According to authors of that paper previously only data from one modality have been taken into consideration for training neural networks to solve that problem. In the example presented in Figure 6.1, there are two modalities: one is related to audio and second to image input which were gathered from video. With the use of data provided into presented autoencoder neural network architecture, [87], a shared representation, in that case the third layer, is obtained. The paper [95] reports superior results of shared representation coming from multiple modalities compared to that with only one modality. However, the results were obtained for a specific problem not related to software bug reports and even similar use in their field of application with the use of different data sources may lead to opposite conclusions. An application of Multimodal Deep Learning in the context other than audio and image application is shown in [136]. It is used for data related to citation networks. It is worth to note that the number of inputs and outputs of artificial neural networks are different in drawings and calculations. The figures clearly present the network with a small number of nodes but in computations the number of inputs and outputs of artificial neural networks is identical with input features and output classes, respectively. Multimodal neural networks differ from others in such a way they have multiple inputs to the network followed by separated parts of at least one hidden layer. It is clearly visualized in Figures 6.1 and 6.3. The concept of multimodal neural networks can be applied when the inputs come from data of different nature. In this study are discussed diverse types of data like categorical, short part of textual data in the form of natural language, as well as log content which is not a standard natural language. Typical is the usage when is processed video, then the data come from both audio and image. However, there are different applications of such neural networks, part of them is presented in Section 1.7.6.

From the point of view of a software development company, it is important that the software bug handling process is as efficient as possible. It is important to continuously develop the tools used to suggest the group in charge for further analysis. One of the ideas is to verify the possibility of introducing multimodal neural network. Research activity related to improvements of software bug assignment are very important for the company. Part of them may end with conclusions that the verified solution which is being verified overbeat the state-of-the-art one. Of course, like in almost all research activities the results are not guaranteed to be positive. Negative results reporting and publishing is also important to show other researchers not to follow the chosen path. Therefore, the aim of such works is to investigate the impact of chosen methods on results over reference methods. One of them used shallow dense (fully connected) neural network with inputs related to title and description of software bug report represented with the use of TF-IDF and categorical data with hot encoding. It is, to some degree, similar to logistic regression with data represented with TF-IDF related to title, description and product type in [38], therefore is threatened as reference state-of-the-art solution. In case of positive outcomes of such activities there might be a recommendation to introduce them in a company. The research related to multimodal network architectures is especially important since it deals with input of diverse input of data. That situation is related to data which the company possesses. It is a mix of natural language text fields like title and description, categorical fields, and data extracted from archives with log content. The architecture of multimodal neural network may vary

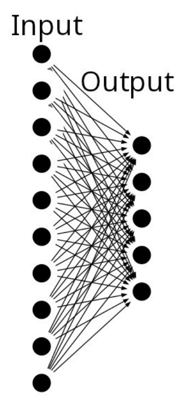


Fig. 6.2 Example of dense (fully connected) neural network.

depending on for instance input data types used, its representations, number of layers.

This paper discusses the following research question:

- RQ.B.1: What is an impact of introducing multimodal neural network on bug assignment accuracy?

Hypothesis:

There exists a method for automated assignment of a software bug report to appropriate development group, responsible for resolving the bug, which outperforms well-known methods for bug report assignment.

6.2 Methods

In this study numerous computational experiments by using different datasets as inputs as well as different architectures of artificial neural networks were performed. The output classes were groups where the cases were finally fixed. The proposed method utilizes a neural network with diverse input types coming from different modalities. To create such a solution there was a need to design the architecture of network specifically to solve that problem and propose how input data would be represented. The following diverse input types, which are used as parts of input layers, or as a whole layer if only one input type is utilized:

ⓐ – part of input layer related to combined text from title and description has been taken. This is a short form in natural language form that describes the discrepancy in the behavior of system. On that data, the preprocessing operations were involved letter changing from upper to lower case and removing special characters, stop words, and email addresses. Lemmatization was also applied. As vectorization TF-IDF was used with n-grams in length range from 1 to 3 and maximum number of features limited to 32 000. Parameters were set using work [38];

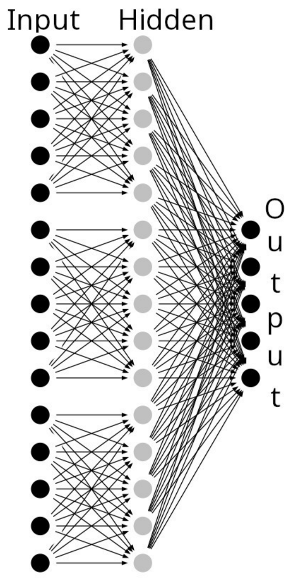


Fig. 6.3 Multimodal deep neural network with labelled Input, Hidden and Output layers. According to [136] when autoencoder architecture is applied then first layer is called input followed by preprocessing layer and the last here is representation layer of autoencoder.

ⓑ – part of input layer was defined by one hot encoded features related to different categorical fields like software release, problem type, product information, repeatability of issue, the phase of cycle where the problem has been discovered;

ⓒ – part of input layer related to content of log archives (snapshots). Their processing used log analyzers which were written as plugins in dedicated platform. Each analyzer was created based on experts' domain knowledge. At least one plugin used for computational experiments was directly related to Radio Frequency (RF) software and Radio Modules. The aim of the current tools is to get the most important log messages and present their content to users with usage of graphical user interface. In this novel approach that information, after being preprocessed and vectorized as described in point ⓐ was used as part of input data to solve the problem of bug report assignment.

The list of experiment of types used in this study includes:

- T1: Artificial neural network (ANN) with the architecture like in Figure 6.4 with input layer with ⓐ.
- T2: ANN with the architecture like in Figure 6.5 with input layer with ⓑ
- T3: ANN with the architecture like in Figure 6.6 with input layer with ⓒ
- T4: ANN with the architecture like in Figure 6.7 with input layer with ⓐ, ⓑ.
- T5: ANN with the architecture like in Figure 6.8 with input layer with ⓐ, ⓑ, ⓒ.
- T6: ANN with the architecture like in Figure 6.9 with input layer with ⓐ, ⓑ.
- T6A: ANN with the architecture like in Figure 6.10 with input layer with ⓐ, ⓑ.

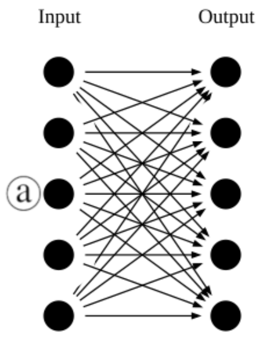


Fig. 6.4 Architecture of neural network T1 with the inputs (a).

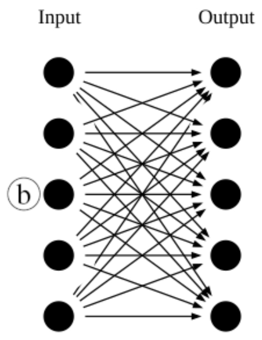


Fig. 6.5 Architecture of neural network T2 with the inputs (b).

- T7: ANN with the architecture like in Figure 6.11 with input layer with ⓐ, ⓑ, ⓒ.
- T7A: ANN with the architecture like in Figure 6.12 with input layer with ⓐ, ⓑ, ⓒ.

Training for experiments named T6A is divided into two phases. In the first one, Autoencoder is trained. For instance, for T6A required Autoencoder architecture is like in Figure 6.1 (please note that in this Figure the input descriptions differ from expected and are demonstrated only for the purpose of visualization of architecture). Then the wages are copied to the network from Figure 6.10 up to the shared representation Layer. Copied wages are then frozen, and the network is fit for obtaining the rest of parameters. Analogous situation is for experiment T7A.

Neural networks were trained with the use of Adam optimizer. In each computational experiment three models were saved: final model derived during training phase, the one with the lowest validation loss, and the highest validation accuracy. Every layer of neural networks dropout has been set to 25%. The activation function softmax was always used in the last layer and ReLu in all other layers.

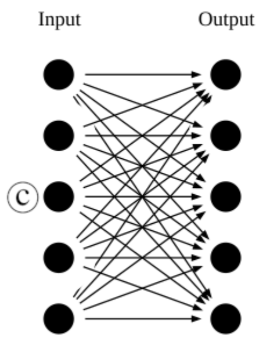


Fig. 6.6 Architecture of neural network T3 with the inputs (c).

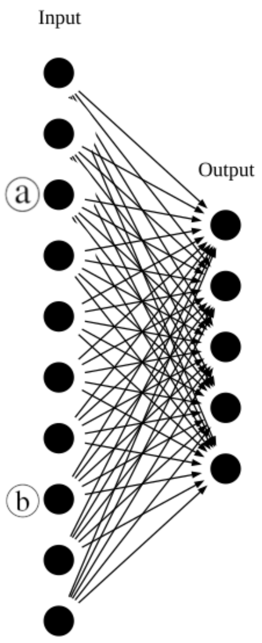


Fig. 6.7 Architecture of neural network T4 with the inputs (a), (b).

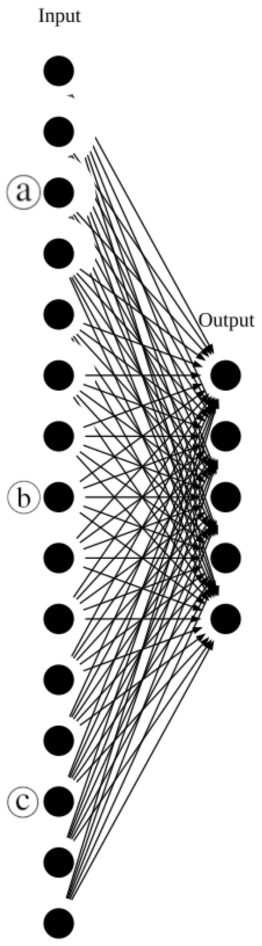


Fig. 6.8 Architecture of neural network T5 with the inputs (a), (b), (c).

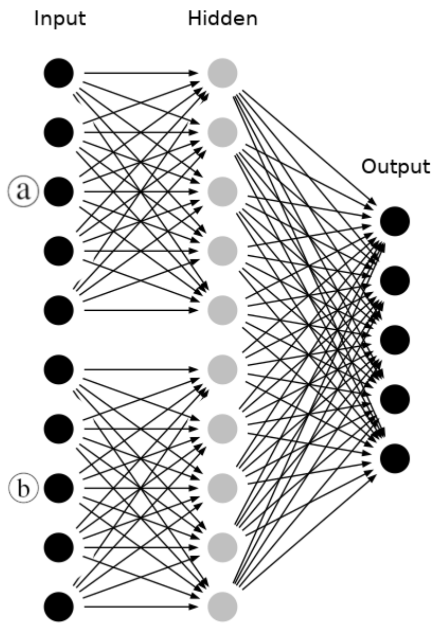


Fig. 6.9 Architecture of neural network T6 with the inputs (a), (b).

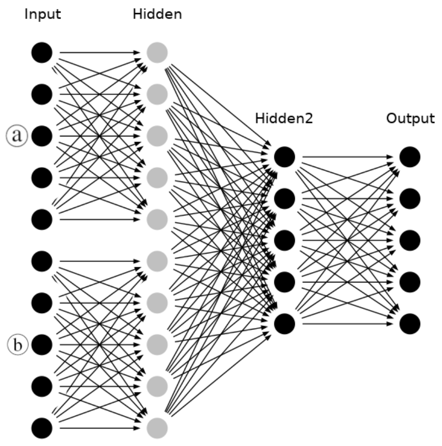


Fig. 6.10 Architecture of neural network T6A with the inputs (a), (b).

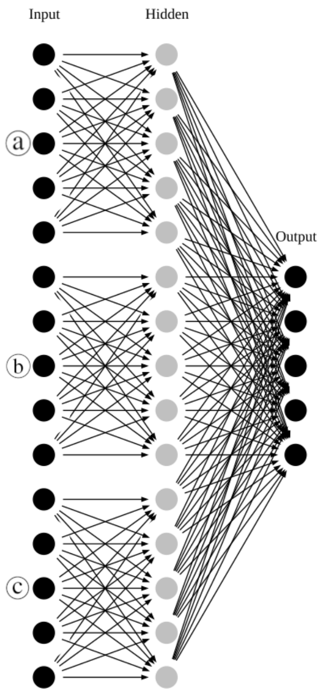


Fig. 6.11 Architecture of neural network T7 with the inputs (a), (b), (c).

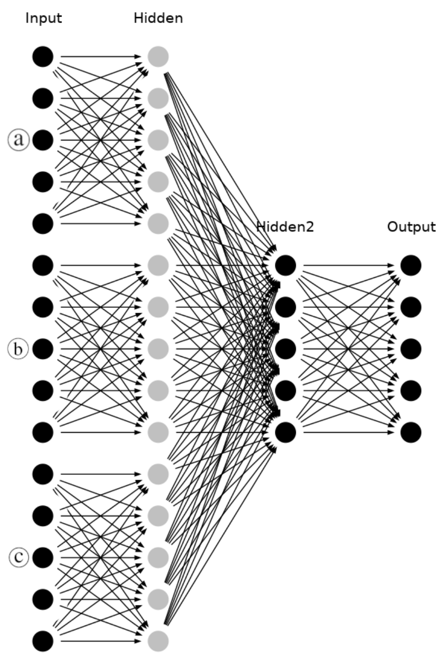


Fig. 6.12 Architecture of neural network T7A with the inputs (a), (b), (c).

Table 6.1 Data quantity related to first series

Train set	Test set	Spare data
7142	359	490

First series

Data with cases which were opened after October 1, 2021 and resolved before May 1, 2022 were used for training. As a test set data related to software bug reports opened after May 1, 2022 and resolved before May 30, 2022 were used. Detailed information about training and test sets quantity are shown in Table 6.1. The spare data shows the quantity of cases reported before May 1, 2022 and resolved after that date. These spare data do not meet requirements to be included in either training or test set. These kind of time dependencies have been introduced in paper [40]. Tests were performed with experiments of type T4 and type T6. In experiment of type T6, the size of each part of hidden layer connected to respective input is equal to the number of neurons of output layer.

Second series

The quantity of data is shown in Table 6.2. The split date for training and test sets was set Feb 1, 2022. For experiment of types T6 and T7 the size of hidden layer parts related to ⓐ, ⓑ, ⓒ is equal to 128, 32, and 128, respectively.

Table 6.2 Data quantity related to second series

Train set	Test set	Spare data
227	72	40

Table 6.3 Data quantity related to third series

Train set	Test set	Spare data
221	330	36

Third series

The quantity of data is shown in Table 6.3. The split date for training and test data was set on Feb 1, 2022. In series 1 and 2 the value of the maximum number of features in TF-IDF was set to 32 000 on inputs to ⓐ and ⓒ. In series 3 and 4 this value was set to every number in the sequence 256, 512, 1024, 2048, 4096, 8192, 16384, 32768. In experiments of types T6 and T7 the sizes of hidden layer parts related to ⓐ, ⓑ, ⓒ are 128, 32, and 128, respectively. Experiments of types T6A and T7A are trained in two phases. In the first phase, Autoencoder is fitted with the sizes of parts of hidden layers the same as the ones defined in experiment of type T6. There are 256 cells used for hidden layer for shared representation. In the next phase, the weights are transferred, and the neural network is finally being fitted with an input layer followed by two layers from Autoencoder and the output layer. After transferring weights, they are no longer being updated during the training of final neural network.

Fourth series

The difference between this and the previous series is the date set in the data split method. The quantity of data is shown in Table 6.4. The split date for training and test sets was set on Jul 1, 2022.

6.3 Results and discussion

First series

Table 6.5 presents the summary of results obtained with different experiment of types and different states of network which were used for the comparison. It seems to be the best to compare different experiment types between each other with the use of the state

Table 6.4 Data quantity related to fourth series

Train set	Test set	Spare data
445	109	33

Table 6.5 Results related to first series of experiment

Experiment type	Accuracy of neural network with best validation loss	Accuracy of neural network with best validation accuracy	Accuracy of last in training neural network
T4	67.97 %	67.97 %	69.64 %
T6	67.97 %	68.25 %	67.41 %

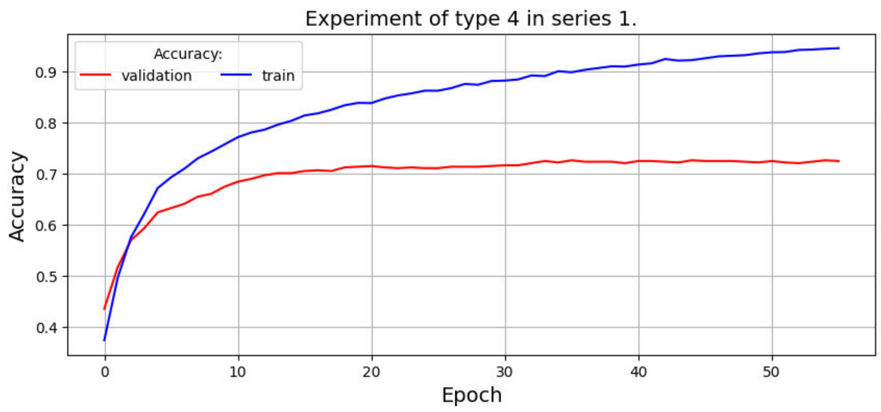


Fig. 6.13 Training and validation accuracy as a function of epoch for experiment of type T4 in series 1.

of neural network taken from early stopping related to parameter of validation loss or validation accuracy. There are also results presented with the last state of network during training phase. In those cases where state of network was taken based on validation loss the prediction results for experiments of types T4 and T6 were identical. Additionally Figure 6.13 and Figure 6.14 show the progress of training of neural networks.

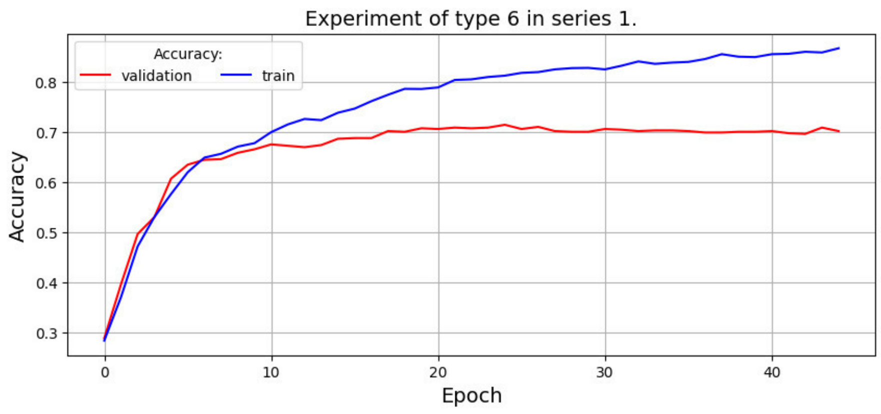


Fig. 6.14 Training and validation accuracy as a function of epoch for experiment of type T6 in series 1.

Table 6.6 Results related to second series of experiment

Experiment type	Accuracy of neural network with best validation loss	Accuracy of neural network with best validation accuracy	Accuracy of last in training neural network
T1	23.61 %	20.83 %	23.61 %
T2	18.06 %	18.06 %	18.06 %
T3	27.78 %	27.78 %	27.78 %
T4	22.22 %	22.22 %	22.22 %
T5	26.39 %	23.61 %	26.39 %
T6	25.00 %	25.00 %	25.00 %
T7	29.17 %	25.00 %	23.61 %

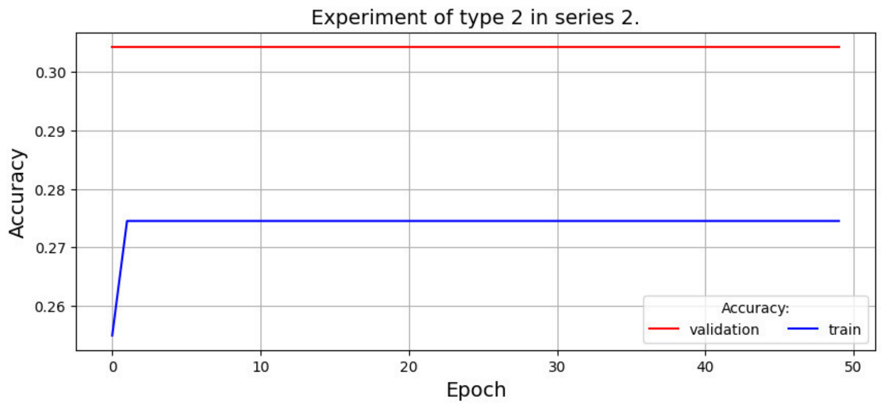


Fig. 6.15 Training and validation accuracy as a function of epoch for experiment of type 2 in series 2.

Second series

Results related to the second series are shown in Table 6.6. When the state of network comes from model checkpoint monitoring validation loss then the highest prediction accuracy among all experiment types in this series is achieved with multimodal architecture of network. It can be clearly noticed that they are better than results obtained with same inputs, but with different architecture of neural network which is fully connected and without hidden layer. Using all types of inputs revealed that the prediction accuracy in experiment of type T7 was higher than for any experiment of types T1-T3. It was observed that providing additional inputs do not always have a positive impact on accuracy. Even if results may be criticized due to overfitting which can be easily noticed based on charts showing accuracy and validation loss as a function of epoch of training (Figures 6.15 and 6.16), they are still relevant. Table 6.6 presents that multimodal neural network architecture gave better results than other ones.

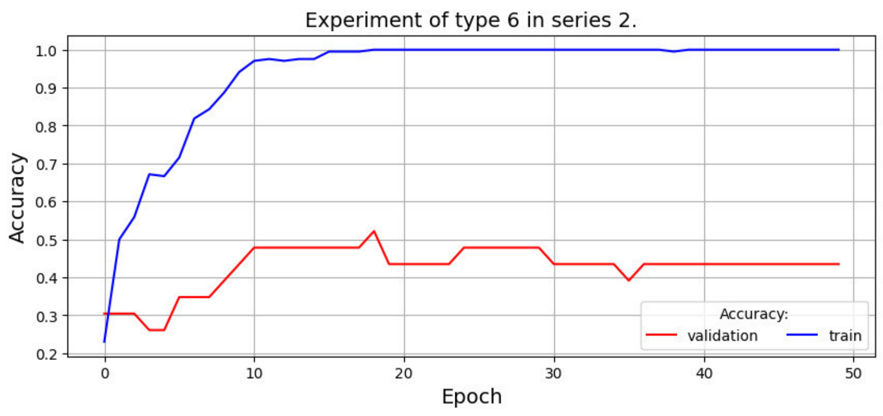


Fig. 6.16 Training and validation accuracy as a function of epoch for experiment of type 6 in series 2.

Third series

Results related to the third series are shown in Table 6.7. Figure 6.17 shows that the highest prediction accuracy was achieved with the use of multiple inputs, but not always with all of them. For part of sub-series related to size of maximum number of features set in TF-IDF, the best result was obtained using multimodal neural network and training with Autoencoder. Figures 6.18 to 6.20, 6.22, 6.23 and 6.25 show accuracy and validation loss as a function of epoch of training. Figures 6.21 and 6.24 present loss and validation loss as a function of epoch of experiment of type T6A for training Autoencoder.

Table 6.7 Results related to third series of experiment

maximum number of features set in TF-IDF	Experiment type	Accuracy of neural network with best validation loss	Accuracy of neural network with best validation accuracy	Accuracy of last in training neural network
256	T1	33.33 %	26.36 %	33.33 %
256	T2	33.64 %	33.64 %	33.64 %
256	T3	29.09 %	22.12 %	29.09 %
256	T4	32.42 %	9.70 %	32.42 %
256	T5	30.30 %	9.39 %	30.30 %
256	T6	33.03 %	33.03 %	31.82 %
256	T6A	35.45 %	34.55 %	35.45 %
256	T7	32.12 %	27.27 %	27.27 %
256	T7A	33.94 %	8.18 %	33.94 %
512	T1	32.73 %	19.39 %	32.73 %

Continued on next page

Table 6.7 – continued from previous page

maximum number of features set in TF-IDF	Experiment type	Accuracy of neural network with best validation loss	Accuracy of neural network with best validation accuracy	Accuracy of last in training neural network
512	T2	33.64 %	33.64 %	33.64 %
512	T3	25.76 %	26.36 %	25.76 %
512	T4	33.03 %	11.82 %	33.03 %
512	T5	33.64 %	29.39 %	29.39 %
512	T6	33.64 %	29.39 %	29.39 %
512	T6A	33.94 %	33.03 %	33.94 %
512	T7	30.30 %	30.61 %	29.70 %
512	T7A	33.33 %	13.03 %	33.33 %
1024	T1	31.82 %	33.64 %	31.82 %
1024	T2	33.64 %	33.64 %	33.64 %
1024	T3	27.58 %	28.48 %	27.58 %
1024	T4	32.12 %	26.67 %	31.52 %
1024	T5	33.94 %	34.24 %	33.33 %
1024	T6	32.42 %	30.30 %	29.39 %
1024	T6A	34.85 %	33.94 %	34.85 %
1024	T7	30.91 %	25.45 %	33.64 %
1024	T7A	33.94 %	33.94 %	33.94 %
2048	T1	31.82 %	16.06 %	31.82 %
2048	T2	33.64 %	33.64 %	33.64 %
2048	T3	28.18 %	30.91 %	28.18 %
2048	T4	31.82 %	26.06 %	31.82 %
2048	T5	33.33 %	23.94 %	33.33 %
2048	T6	32.73 %	33.94 %	30.91 %
2048	T6A	32.73 %	21.82 %	32.73 %
2048	T7	34.55 %	31.52 %	34.24 %
2048	T7A	34.85 %	33.64 %	34.85 %
4096	T1	31.21 %	31.21 %	31.21 %
4096	T2	33.64 %	33.64 %	33.64 %
4096	T3	28.79 %	28.79 %	28.79 %
4096	T4	31.21 %	23.33 %	31.21 %
4096	T5	33.64 %	27.58 %	34.24 %
4096	T6	32.42 %	34.55 %	26.36 %

Continued on next page

Table 6.7 – continued from previous page

maximum number of features set in TF-IDF	Experiment type	Accuracy of neural network with best validation loss	Accuracy of neural network with best validation accuracy	Accuracy of last in training neural network
4096	T6A	34.24 %	33.94 %	34.24 %
4096	T7	35.15 %	32.73 %	32.73 %
4096	T7A	33.94 %	34.24 %	33.94 %
8192	T1	31.21 %	33.94 %	31.21 %
8192	T2	33.64 %	33.64 %	33.64 %
8192	T3	28.48 %	28.48 %	28.48 %
8192	T4	31.52 %	30.91 %	31.52 %
8192	T5	33.94 %	31.82 %	33.94 %
8192	T6	32.12 %	34.24 %	30.30 %
8192	T6A	33.94 %	33.33 %	33.94 %
8192	T7	36.67 %	33.64 %	32.73 %
8192	T7A	33.94 %	33.64 %	33.94 %
16384	T1	31.52 %	33.64 %	31.52 %
16384	T2	33.64 %	33.64 %	33.64 %
16384	T3	29.09 %	33.94 %	29.39 %
16384	T4	31.82 %	31.52 %	31.82 %
16384	T5	34.24 %	33.94 %	34.24 %
16384	T6	33.64 %	30.91 %	30.00 %
16384	T6A	34.55 %	11.52 %	34.55 %
16384	T7	35.76 %	21.21 %	34.55 %
16384	T7A	33.94 %	12.42 %	33.94 %
32768	T1	31.82 %	33.33 %	31.82 %
32768	T2	33.64 %	33.64 %	33.64 %
32768	T3	28.48 %	28.48 %	28.48 %
32768	T4	31.52 %	33.64 %	31.52 %
32768	T5	33.33 %	34.85 %	33.33 %
32768	T6	33.03 %	33.94 %	30.61 %
32768	T6A	34.55 %	33.64 %	34.55 %
32768	T7	36.97 %	35.15 %	34.55 %
32768	T7A	33.94 %	23.64 %	33.94 %

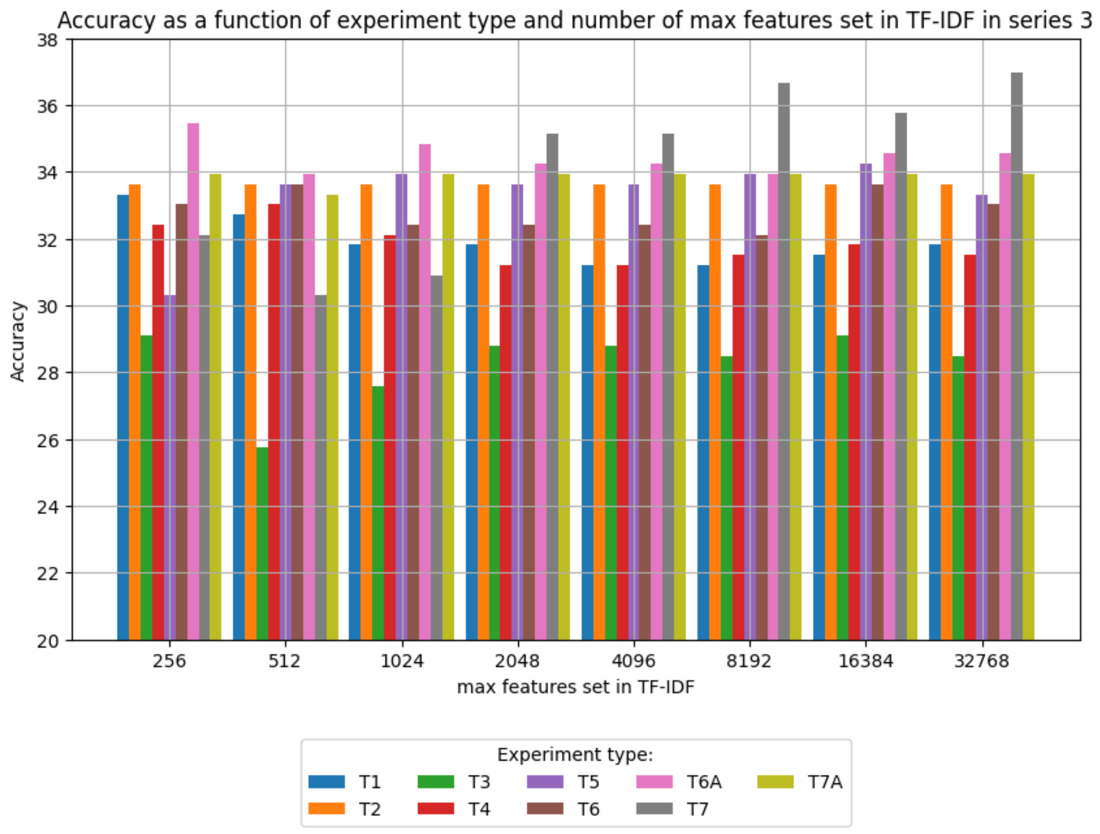


Fig. 6.17 Accuracy as a function of experiment of type and number of maximum number of features set in TF-IDF in series 3. Presented accuracy was calculated with the state of network from model checkpoint monitoring validation loss.

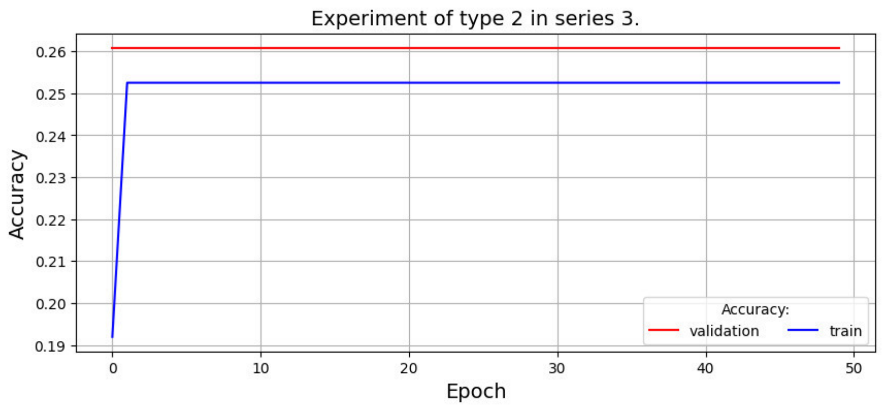


Fig. 6.18 Training and validation accuracy as a function of epoch for experiment of type T2 in series 3.

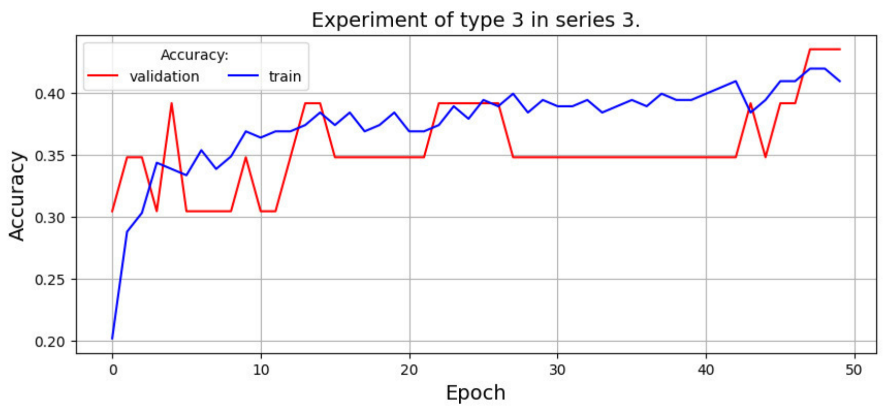


Fig. 6.19 Training and validation accuracy as a function of epoch for experiment of type T3 in series 3.

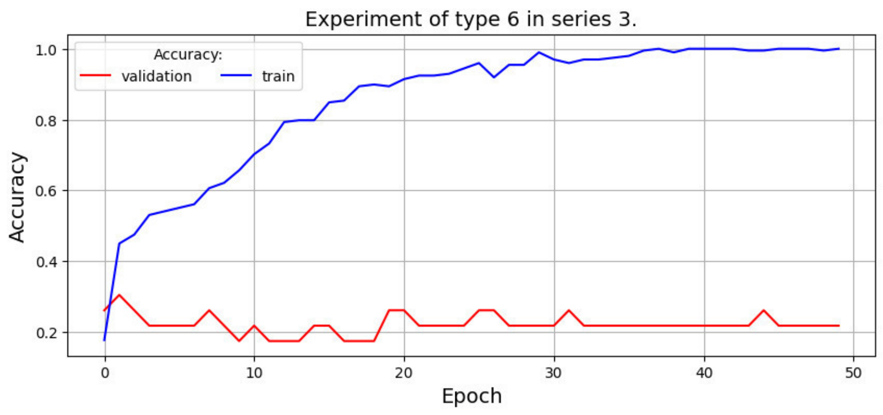


Fig. 6.20 Training and validation accuracy as a function of epoch for experiment of type T6 in series 3.

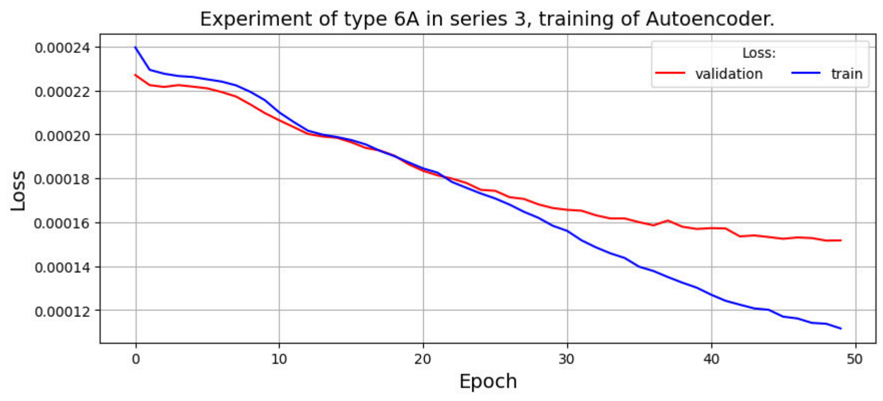


Fig. 6.21 Loss and validation loss as a function of epoch for experiment of type T6A in series 3 for training Autoencoder.

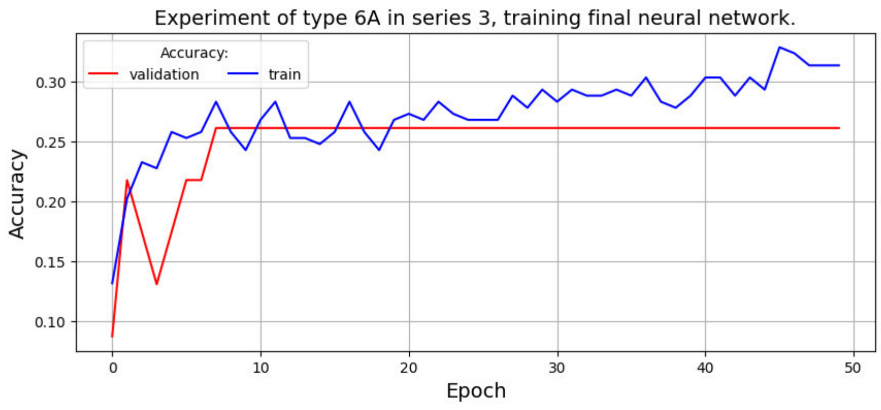


Fig. 6.22 Training and validation accuracy as a function of epoch for experiment of type T6A in series 3 for training classification model.

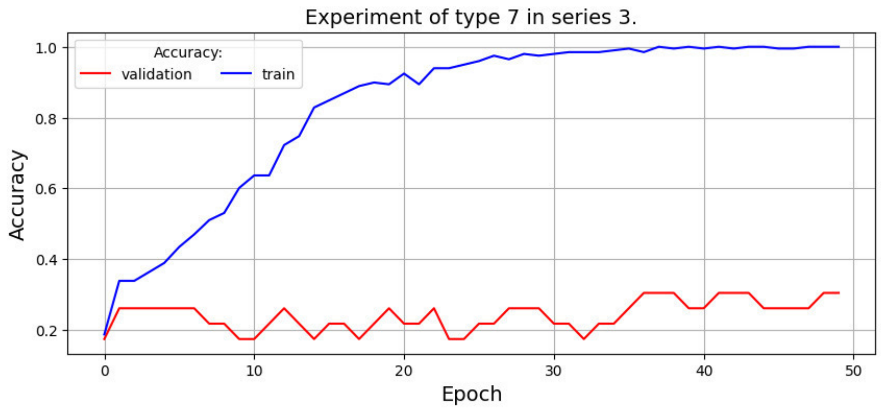


Fig. 6.23 Training and validation accuracy as a function of epoch for experiment of type T7 in series 3.

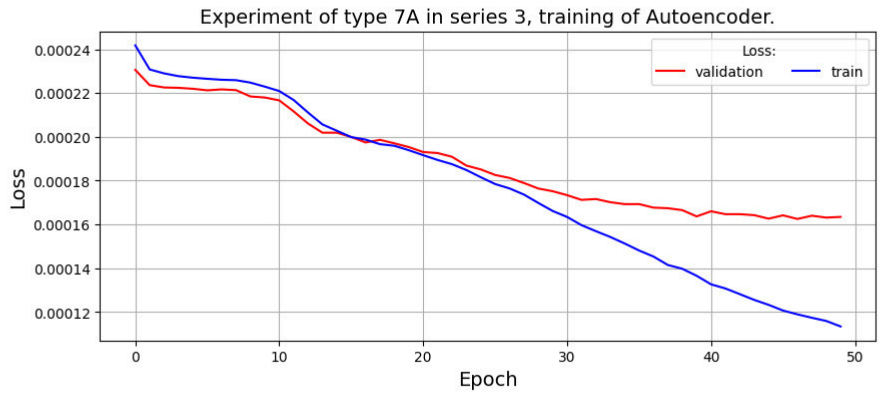


Fig. 6.24 Loss and validation loss as a function of epoch for experiment of type T7A in series 3 for training Autoencoder.

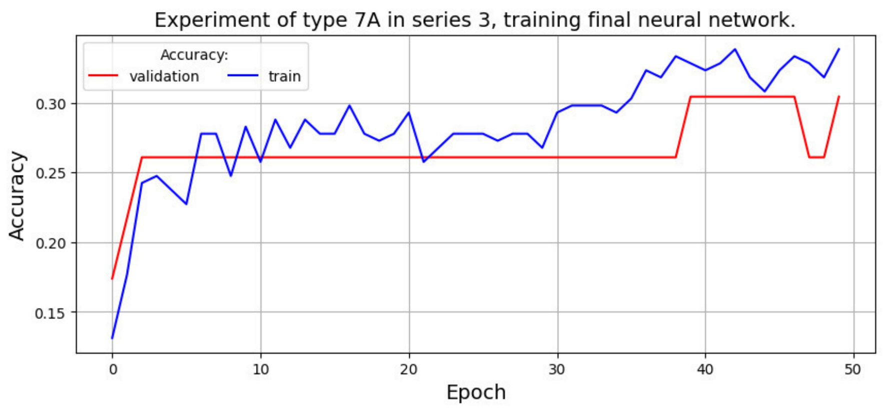


Fig. 6.25 Training and validation accuracy as a function of epoch for experiment of type T7A in series 3 for training classification model.

Fourth series

Results related to the fourth series are shown in Table 6.8. Figure 6.26 shows that the highest prediction accuracy was achieved with experiment of type T3 (used single input ⓒ). Given multiple inputs the highest prediction accuracy, in most cases, was achieved by Autoencoder used to train multimodal neural network. Figures 6.27 and 6.29 show accuracy and validation loss as a function of epoch of training. Figure 6.28 present loss and validation loss as a function of epoch of experiment of type T6A for training Autoencoder.

Table 6.8 Results related to fourth series of experiment

maximum number of features set in TF-IDF	Experiment type	Accuracy of neural network with best validation loss	Accuracy of neural network with best validation accuracy	Accuracy of last in training neural network
256	T1	33.03 %	38.53 %	33.03 %
256	T2	37.61 %	37.71 %	37.61 %
256	T3	38.53 %	33.94 %	38.53 %
256	T4	34.86 %	34.86 %	34.86 %
256	T5	35.78 %	38.53 %	35.78 %
256	T6	28.44 %	28.44 %	28.44 %
256	T6A	36.70 %	39.54 %	36.70 %
256	T7	33.94 %	30.28 %	31.19 %
256	T7A	35.78 %	22.94 %	35.78 %
512	T1	36.70 %	36.70 %	36.70 %
512	T2	37.61 %	37.61 %	37.61 %

Continued on next page

Table 6.8 – continued from previous page

maximum number of features set in TF-IDF	Experiment type	Accuracy of neural network with best validation loss	Accuracy of neural network with best validation accuracy	Accuracy of last in training neural network
512	T3	39.45 %	39.45 %	39.45 %
512	T4	36.70 %	34.86 %	36.70 %
512	T5	33.03 %	35.78 %	33.03 %
512	T6	34.86 %	32.11 %	30.28 %
512	T6A	39.45 %	37.61 %	39.45 %
512	T7	28.44 %	31.19 %	30.28 %
512	T7A	37.61 %	36.70 %	37.61 %
1024	T1	33.94 %	34.86 %	33.94 %
1024	T2	37.61 %	37.61 %	37.61 %
1024	T3	39.45 %	31.19 %	39.45 %
1024	T4	34.86 %	35.78 %	34.86 %
1024	T5	33.94 %	40.37 %	33.94 %
1024	T6	35.78 %	33.94 %	33.03 %
1024	T6A	37.61 %	32.11 %	37.61 %
1024	T7	33.03 %	30.28 %	35.78 %
1024	T7A	37.61 %	37.61 %	37.61 %
2048	T1	32.11 %	34.86 %	32.11 %
2048	T2	37.61 %	37.61 %	37.61 %
2048	T3	37.61 %	37.61 %	37.61 %
2048	T4	33.03 %	33.03 %	33.03 %
2048	T5	31.19 %	38.53 %	31.19 %
2048	T6	36.70 %	27.52 %	28.44 %
2048	T6A	35.78 %	32.11 %	35.78 %
2048	T7	34.86 %	27.52 %	29.36 %
2048	T7A	39.45 %	39.45 %	39.45 %
4096	T1	33.94 %	36.70 %	33.94 %
4096	T2	37.61 %	37.61 %	37.71 %
4096	T3	38.53 %	38.53 %	38.53 %
4096	T4	32.11 %	33.94 %	32.11 %
4096	T5	30.28 %	30.28 %	30.28 %
4096	T6	37.61 %	36.70 %	35.78 %
4096	T6A	36.70 %	37.61 %	36.70 %

Continued on next page

Table 6.8 – continued from previous page

maximum number of features set in TF-IDF	Experiment type	Accuracy of neural network with best validation loss	Accuracy of neural network with best validation accuracy	Accuracy of last in training neural network
4096	T7	38.53 %	35.78 %	33.03 %
4096	T7A	37.61 %	14.68 %	37.61 %
8192	T1	32.11 %	30.28 %	32.11 %
8192	T2	37.61 %	13.76 %	37.61 %
8192	T3	39.45 %	39.45 %	39.45 %
8192	T4	32.11 %	36.70 %	32.11 %
8192	T5	33.03 %	33.03 %	33.03 %
8192	T6	35.78 %	33.03 %	36.70 %
8192	T6A	37.61 %	20.18 %	37.71 %
8192	T7	33.03 %	36.70 %	35.78 %
8192	T7A	37.61 %	26.61 %	37.61 %
16384	T1	32.11 %	31.19 %	32.11 %
16384	T2	37.61 %	37.61 %	37.61 %
16384	T3	40.37 %	41.28 %	40.37 %
16384	T4	31.19 %	32.11 %	31.19 %
16384	T5	33.03 %	32.11 %	33.03 %
16384	T6	33.94 %	33.03 %	34.86 %
16384	T6A	37.61 %	37.60 %	37.61 %
16384	T7	34.86 %	35.78 %	36.70 %
16384	T7A	36.70 %	36.70 %	36.70 %
32768	T1	32.11 %	33.03 %	32.11 %
32768	T2	37.61 %	37.61 %	37.61 %
32768	T3	38.53 %	38.53 %	39.45 %
32768	T4	31.19 %	33.03 %	31.19 %
32768	T5	33.03 %	32.11 %	33.03 %
32768	T6	35.78 %	34.86 %	36.70 %
32768	T6A	39.45 %	39.45 %	39.45 %
32768	T7	34.86 %	37.61 %	35.78 %
32768	T7A	37.61 %	14.68 %	37.61 %

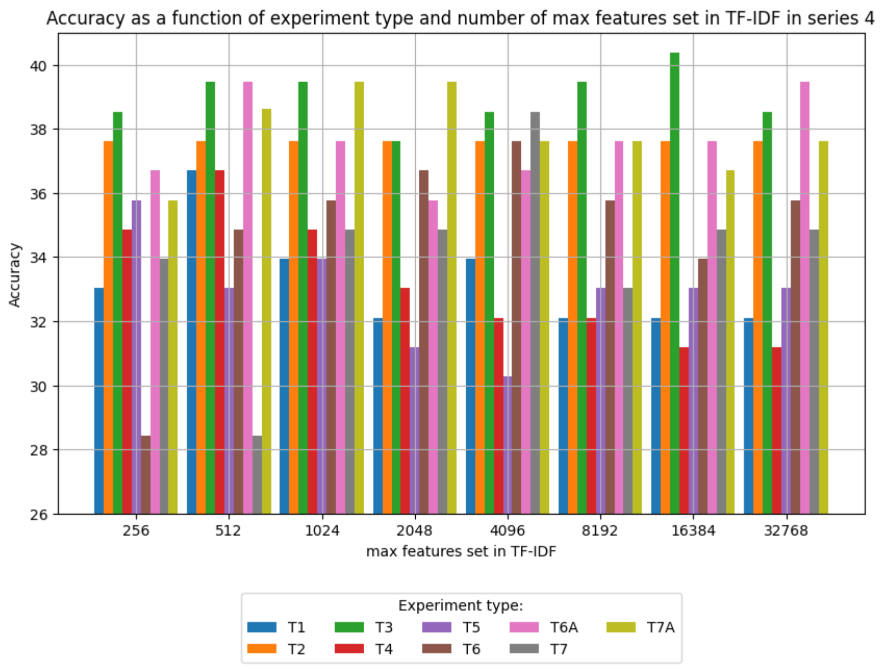


Fig. 6.26 Accuracy as a function of experiment of type and number of maximum number of features set in TF-IDF in series 4.

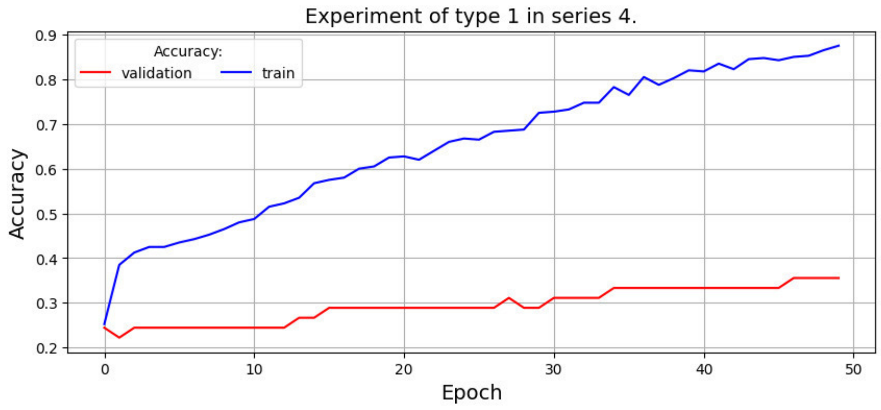


Fig. 6.27 Training and validation accuracy as a function of epoch for experiment of type T1 in series 4.

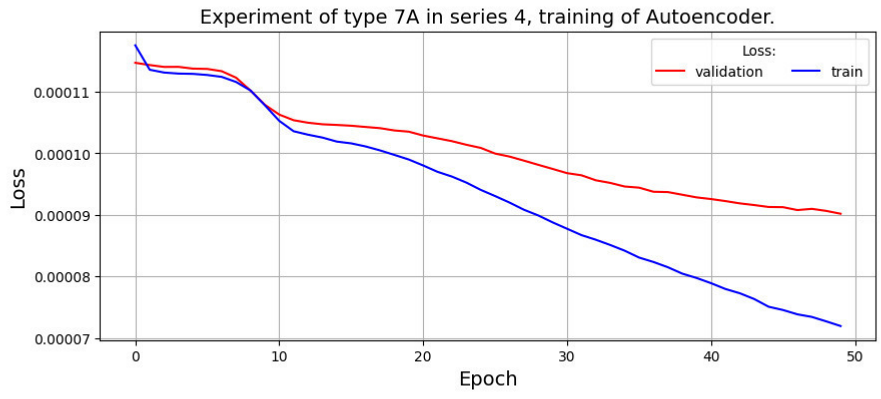


Fig. 6.28 Loss and validation loss as a function of epoch for experiment of type T7A in series 4 for training Autoencoder.

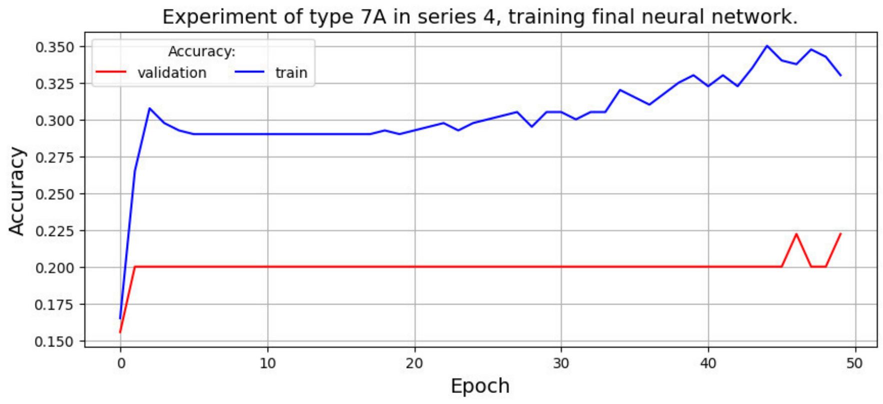


Fig. 6.29 Training and validation accuracy as a function of epoch for experiment of type T7A in series 4 for training classification model. Presented accuracy was calculated with the state of network from model checkpoint monitoring validation loss.

Discussions

There are some limitations of the presented approach especially at the current state as they are related to the data used in this study. The constraints involve situations where not all problem reports contain data required to correctly extract all modalities. To overcome this difficulty, only sets without missing data were used in this study. It should be noted that in the series in which there was a greater cardinality of the sets, the results obtained were higher. The novel approach used only a part of internal company data and it was compared with state-of-the-art methods. The utilization of data coming from different modalities showed that there was room for model prediction improvement. This observation is a significant element related to software bug report assignments. It shows that with the use of different data than typical like from title and description of issue the results may be achieved higher, but at the same time we must remember that such solution is not immune to log changes in the time and such changes would affect the results.

6.4 Conclusions

The paper presents different approaches of assigning software bug reports to appropriate investigation or development group with the use of machine learning techniques. A few key findings are summarized in this section.

The paper compares results which can be obtained with the use of diverse parts from available data about bug reports. These parts involve natural language processing applied to title and description, categorical fields, and log archives from which features are further extracted. Data from snapshots are processed with the use of solution which aims to get most important log messages. Next, the processing of such data extracted from log messages includes operations like changing letters from upper to lower, removing stopwords. The same steps are applied for preprocessing of title and description. Afterwards, TF-IDF was employed and verified with the use of different settings for maximum number of features set. Categorical data were transformed with the use of hot encoding. The analysis of results showed that in general no single data input type can lead to superior model prediction accuracy. The majority of the experiments conducted in this study showed that the highest model accuracy was obtained when archives of logs and/or categorical data were also used for building model. Therefore, it can be concluded that introduced methods improve software bug assignment accuracy and, at same time, confirm the hypothesis that *There exists a method for automated assignment of a software bug report to appropriate development group, responsible for resolving the bug, which outperforms well-known methods for bug report assignment.* It is not clear that the methods introduced will always improve software bug assignment accuracy but it is still a big step in the development of such approaches, and it is still worth to further investigating that topic.

Regarding further answering the research question *about an impact of introducing multimodal neural network on bug assignment accuracy* and confirming about hypothesis there were made comparison with state-of-the-art methods used in company. These reference methods come from [38] and are similar to type T1 and T4. Based on t-test performed for selected experiments from fourth series with null hypothesis verifying whether that two independent samples have identical expected values [121]. It is shown that there is a significant difference between results gained from experiment type T1 and type T7A (p-value 0.00014 for null hypothesis) as well as type T4 and T7A (p-value 0.00048 for null hypothesis) what highlights the importance of results. These results combined with the fact that average value of software bug report assignment accuracy for model with architecture T7A was higher lead to conclusion that the results with the architecture T7A are improved significantly in statistical way.

The analysis showed that neural network architecture with the use of separated hidden layers (see, for example, Figure 6.3) for multiple inputs, in most cases lead to better prediction accuracy than obtained with fully connected neural network (see, for example, Figure 6.2) with respective inputs. This can be observed in Figures 6.17 and 6.26 comparing types of experiments between each other with same setting of maximum features set in TF-IDF. The conducted experiments show the advantages of the use of multimodal neural network in the application of assigning software bug reports to groups of developers or preinvestigators.

In all series of experiments of type 2 with categorical data the maximum value of training and validation accuracy was obtained at early phases of training. No further improvement is seen after third epoch (Figures 6.15 and 6.18), yielding that for this kind of training early stopping criteria might be applied without degradation of model.

Figures 6.21, 6.24 and 6.28 show smooth curves representing loss as a function of epoch during training of Autoencoders. All of these curves have a decreasing trend. The smoothness and decreasing trend confirm correct learning process of the model. They are related to experiments of types 6A and 7A, where training of network was divided into two phases, first related to training of Autoencoder, second related to obtaining the weights in the rest of the network. Experiments of types 6A and 7A result in smaller difference between training and validation accuracy (Figures 6.22, 6.25 and 6.29) comparing to the other experiment types (Figures 6.13 to 6.16, 6.18 to 6.20, 6.23 and 6.27). In the second phase of training there was only one layer fitted after transformation of weights from Autoencoder. It leads to simplification of the prediction model, by employing fewer layers and neurons. This means that fewer weights are required to be fitted. Moreover, results of experiments of experiments of types 6A and 7A have less difference in model prediction accuracy at training and test phases, suggesting the impact of overfitting minimized.

CHAPTER 7

Architecture, Environment and Orchestration

7.1 Introduction

This manual is intended for developers and DevOps engineers of services related to machine learning software bug assignment predictions. It contains also a part related to data gathering and processing. Figure 7.1 presents a general overview of systems architecture which includes Reporting System, Data System (Section 7.2), Model Retraining System (Section 7.4) and Model Serving System (Section 7.3). Reporting System is a main system which is responsible for storing information about software bug reports, whereas Data system is a solution created for the purpose of more convenient access to data. Model Retraining System gathers data from Data System and updates the Model Serving System regularly providing newer models.

7.2 Data system

Data system is located in the namespace `data-services`. General information related to data services is present in Figure 7.2. Reporting System is the main system in company which stores information regarding software bug reports. In Figures 7.3 to 7.5, Reporting System refers to exactly the same entity. Software Bug Report Dataset Builder (Section 7.2.1) gathers data from Reporting System and update dataset in organized manner in Storage. Regularly, there is run a task which performs snapshot processing (Section 7.2.3) on data gathered from Storage. Part of data which is kept in this storage is accessible for further model retraining via Data Selection Service (Section 7.2.2). Moreover, there is a Postgres Service for the purpose of keeping data for PowerBi [10].

7.2.1 Software Bug Report Dataset Builder

Dataset Builder is responsible for gathering data from tool which is utilized in company to keep the data related to software bug reports. The main purpose of those services is to create a local copy of data. It contains information about title, description and

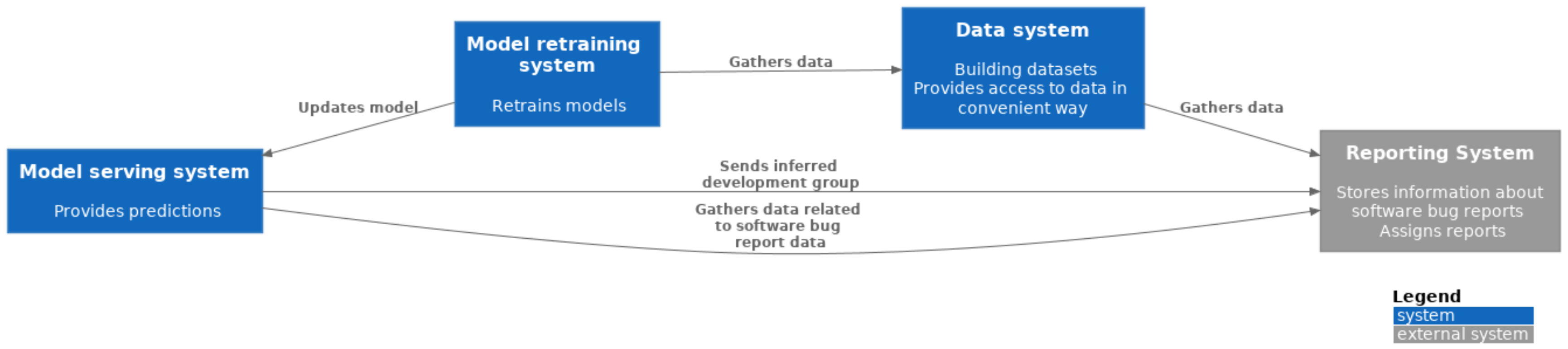


Fig. 7.1 General overview of systems architecture.

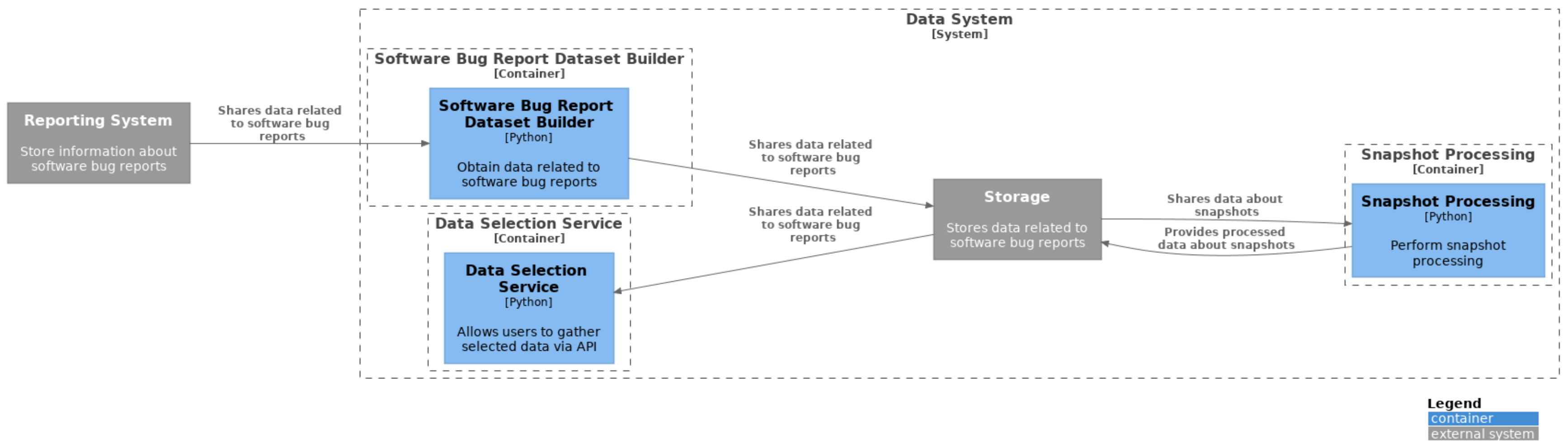

Data System
[System]
Reporting System
Store information about software bug reports
Shares data related to software bug reports
Software Bug Report Dataset Builder
[Container]
Software Bug Report Dataset Builder
[Python]
Obtain data related to software bug reports
Shares data related to software bug reports
Data Selection Service
[Container]
Data Selection Service
[Python]
Allows users to gather selected data via API
Shares data related to software bug reports
Storage
Stores data related to software bug reports
Shares data about snapshots
Provides processed data about snapshots
Snapshot Processing
[Container]
Snapshot Processing
[Python]
Perform snapshot processing
Legend
container
external system

Fig. 7.2 Component diagram which includes data services.

some categorical fields of software bug reports. Description field may contain also information about references to logs or parts of logs themselves.

7.2.2 Data Selection Service

Data Selection Service is a service which allows user to send a request to get selected part of data created with the use of Software Bug Report Dataset Builder described in Section 7.2.1.

7.2.3 Snapshot processing

Snapshots processing combines gathering zip files related to particular cases and analyzing them. The process is performed in Docker [85, 68] and Kubernetes [8] environments.

7.2.4 Postgres service

Snapshots processing combines gathering zip files related to particular cases and analyzing them. The process is performed in Docker [85, 68] and Kubernetes [8] environments.

7.3 Model serving system

7.3.1 General overview of major components

This Section presents three approaches of systems responsible for providing predictions related to software bug assignment with or without using archive of logs. Most of the system is kept in namespace called namespace-production. A log may be defined as an operation system information which is saved to any kind of file including standard Syslog [51] or other specific files containing binary data or even files containing information about configuration. The approach without using archive of logs may employ description of the problem report which also may contain log content. The architecture of the approach with the use of only archive of logs is shown in Figure 7.3. The architecture of the approach without using archive of logs is visualized in Figure 7.4 a similar way with two modifications: remove Analyzer Core (Container) and send suspected groups predictions from "Main service for predictions" directly to Reporting system. There is also visualized combined version in Figure 7.5 which utilizes at once data from archive of logs as well as data from title, description and categorical fields of software bug report.

Currently, Environment and Orchestration of machine learning services related to software bug report assignment are based on two different architectures depending on the data being taken as input. Nevertheless, they have common services to be reused by design to avoid code duplicates and avoid duplicating the same kind of resource if possible. Both architectures, by their design, use the same kind of preprocessing service; if possible, they use the same instance for both. To avoid as much as possible dependencies from systems and python packages with a specific version, Docker [85], was introduced. To provide high availability of service, Kubernetes [119] was employed. The major benefits of these two software management solutions are reliability of services and possibility of creating updates with rolling update policy [9]. That policy enables the option to update the version of services related to machine learning models from currently used for newer without any break in availability of services during that update.

7.3.2 Analyzer core

It is a part responsible among others for reading log archives to extract selected data. As next step, Analyzer core requests main service to get machine learning prediction and deliver it to Reporting system which may decide based on strict rules whether do the transfer of problem report to another team responsible or not (see Figure 7.3). Developers deliver source code available under: repository link, according to standardized internal template and the maintenance of service is under responsibility of authors of respective plugins, not maintainers of main platform.

7.3.3 Main service for predictions

The main services keep the machine learning models running and generate software bug assignment predictions. They use the preprocessing activity described in 7.3.4 during both the production mode and the training phase. By production mode, we understand the mode when the user gets predictions based on the API. They also use the vectorizer function, described in 7.4.1, during the training phase.

7.3.4 Preprocessing service

The preprocessing service cleans data based on the configuration set by the user in the parameters. It applies selected standard preprocessing techniques from the natural language processing field and also specific ones related to company data.

7.3.5 Filtering service

Filtering Service provides filter predictions based on given information. As an example, it removes part of predictions related to particular groups in postprocessing.

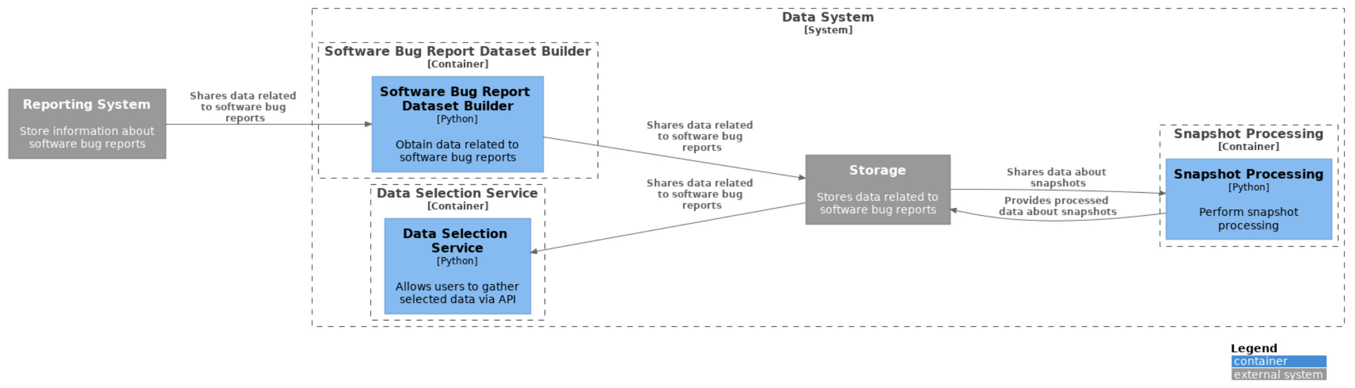

Data System
[System]
Software Bug Report Dataset Builder
[Container]
Software Bug Report Dataset Builder
[Python]
Obtain data related to software bug reports
Reporting System
Store information about software bug reports
Shares data related to software bug reports
Shares data related to software bug reports
Shares data related to software bug reports
Storage
Stores data related to software bug reports
Data Selection Service
[Container]
Data Selection Service
[Python]
Allows users to gather selected data via API
Shares data about snapshots
Provides processed data about snapshots
Snapshot Processing
[Container]
Snapshot Processing
[Python]
Perform snapshot processing
Legend
container
external system

Fig. 7.3 Component diagram of system responsible for providing predictions based on content extracted from archive of logs.

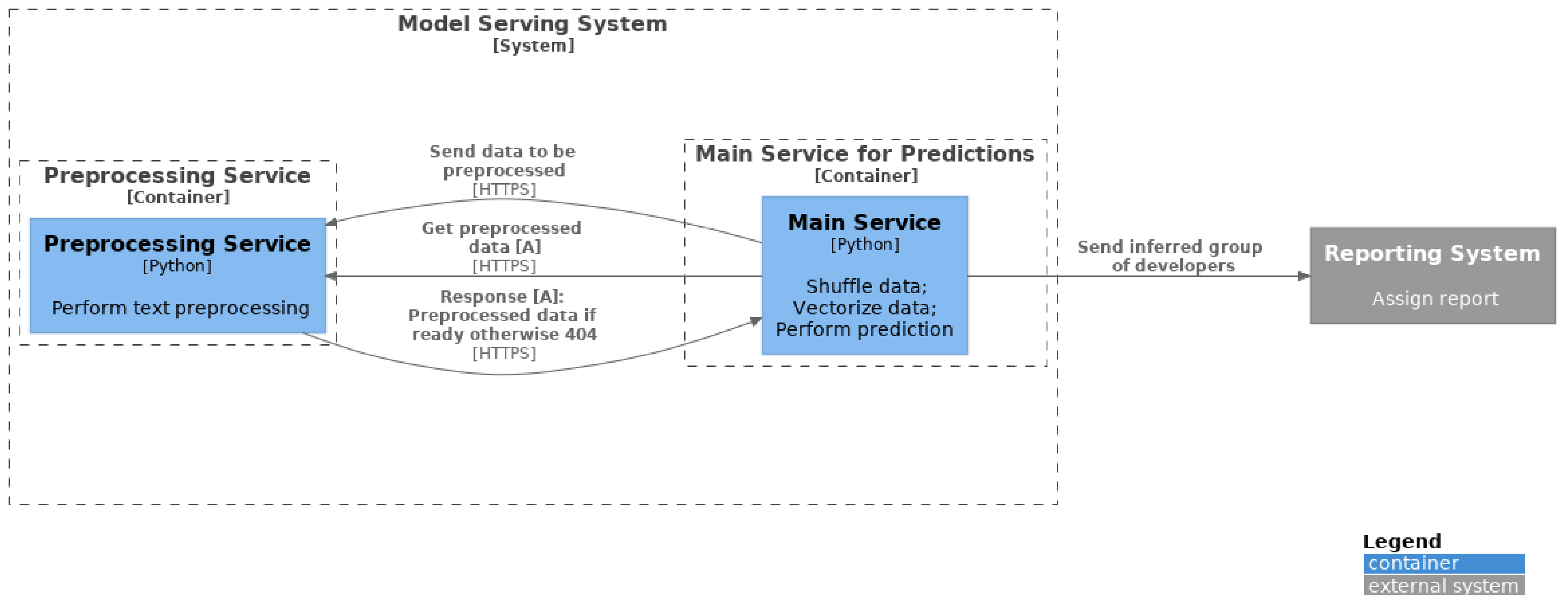
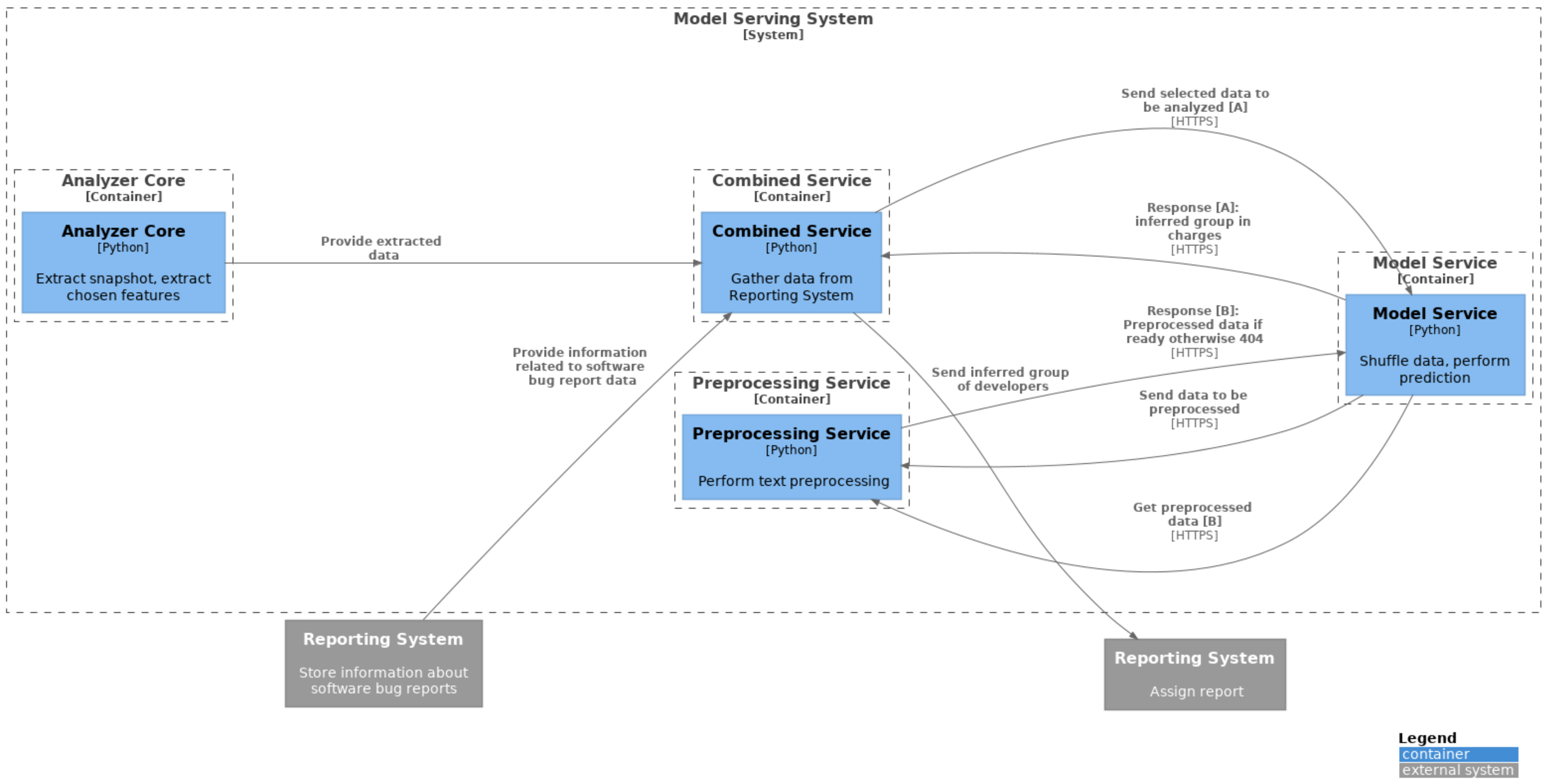

Model Serving System
[System]
Preprocessing Service
[Container]
Preprocessing Service
[Python]
Perform text preprocessing
Send data to be preprocessed
[HTTPS]
Get preprocessed data [A]
[HTTPS]
Response [A]: Preprocessed data if ready otherwise 404
[HTTPS]
Main Service for Predictions
[Container]
Main Service
[Python]
Shuffle data; Vectorize data; Perform prediction
Send inferred group of developers
Reporting System
Assign report
Legend
container
external system

Fig. 7.4 Component diagram of system responsible for providing predictions based on title, description and categorical fields.



Model Serving System
[System]
Analyzer Core
[Container]
Analyzer Core
[Python]
Extract snapshot, extract chosen features
Provide extracted data
Combined Service
[Container]
Combined Service
[Python]
Gather data from Reporting System
Send selected data to be analyzed [A]
[HTTPS]
Response [A]: inferred group in charges
[HTTPS]
Model Service
[Container]
Model Service
[Python]
Shuffle data, perform prediction
Response [B]: Preprocessed data if ready otherwise 404
[HTTPS]
Send inferred group of developers
Send data to be preprocessed
[HTTPS]
Provide information related to software bug report data
Preprocessing Service
[Container]
Preprocessing Service
[Python]
Perform text preprocessing
Get preprocessed data [B]
[HTTPS]
Reporting System
Store information about software bug reports
Reporting System
Assign report
Legend
container
external system

Fig. 7.5 Component diagram of system responsible for providing predictions based on title, description, categorical fields and archive of logs.

7.3.6 Direct hardware unit mapping

Direct Hardware Unit mapping predicts groups only based on information related to the presence of hardware used.

7.3.7 Part of production setup installation

All configuration data except secrets are stored in repository. Chosen requirements are presented below:

7.3.7.1 Required namespace

1. `namespace-production`

7.3.7.2 Required secrets

1. `s3cfg-experiments-archive`
2. `research-services-pass`
3. `apm-private-key`

7.3.7.3 Required services (deployment + service + ingress)

1. `data-preprocessor`
2. `A2_to_A6`
3. `cross_department_predictions`
4. `direct_hwunit_mapping`
5. `internal_predictions`
6. `internal_predictions_analyzer`
7. `internal_predictions_analyzer_prototype_1`
8. `internal_predictions_analyzer_prototype_2`
9. `internal_predictions_analyzer_prototype_3`
10. `internal_predictions_pilot`
11. `predictions_combiner_service`
12. `A6_threshold`
13. `A6_to_A2`

7.3.7.4 Required volumes

1. `data-preprocessor`

7.4 Model Retraining System

For the creation of models for production use research-services namespace is utilized. Configurations related to part of updating of machine learning services is kept in respective repository. Retraining is done with the use of Kubernetes CronJobs. For the purpose of retraining the models is utilized same part of services which are also present in Section 7.3 (Section 7.3.4, Section 7.3.3), but also Section 7.4.1 which is not required in Section 7.3.

7.4.1 Vectorizer service

It is a service responsible for vectorization of preprocessed data. User provides parameters and data. Then the service produces vectorizer or vectorized data. There is also an option to remove the vectorized data from service. The service uses asynchronous communication.

7.5 Installation of SSL certificate

To install a wildcard SSL certificate in ingress by default with .pem and .key files:

- `sample-domain.pem`,
- `sample-domain.key`

use the following steps:

1. convert .pem and .key files to base64 format as shown in Listing 1;
2. modify the template shown in Listing 2 by adding TLS secret to Kubernetes ;
3. edit daemonset related to SSL ingress by adding line with reference to default SSL certificate (Listings 3 and 4).

Listing 1: Encode with base64 SSL certificates.

```
cat sample-domain.pem | base64
LS0tLS1CRUd...
cat sample-domain.key | base64
LS0tLS1CRU...
```

Listing 2: Example of Kubernetes secret with SSL certificate.

```
apiVersion: v1
data:
  tls.crt: LS0tLS1CRUd...
  tls.key: LS0tLS1CR...
kind: Secret
metadata:
  annotations:
    cert-manager.io/alt-names: 'sample-domain.net'
    cert-manager.io/certificate-name: tls-tools
    cert-manager.io/common-name: 'sample-domain.net'
    cert-manager.io/issuer-group: www.digicert.com
    cert-manager.io/issuer-kind: Issuer
    cert-manager.io/issuer-name: DigiCert Inc
  name: tls-sample-domain
  namespace: ingress-nginx
type: kubernetes.io/tls

```

Listing 3: Command to edit ingress NGNIX to set default SSL certificate.

```
kubectl --namespace=ingress-nginx edit
 ↪  daemonset/ingress-nginx-controller
```

Listing 4: Part of configuration of ngnix ingress controller.

```
      containers:
      - args:
        - /nginx-ingress-controller
        - --publish-service=$(POD_NAMESPACE)/
         ↪  ingress-nginx-controller
        - --election-id=ingress-controller-leader
        - --controller-class=k8s.io/ingress-nginx
        - --ingress-class=nginx
        - --configmap= $(POD_NAMESPACE)/ingress-nginx-controller
        - --validating-webhook=:8443
        - --validating-webhook-certificate=
         ↪  /usr/local/certificates/cert
        - --validating-webhook-key= /usr/local/certificates/key
        - --watch-ingress-without-class=true
        - --default-ssl-certificate= ingress-nginx/tls-sample-domain
        env:
        - name: POD_NAME

```

7.6 Clearing not needed images

In order to clear actions related to the removal of images from remote Artifactory, the following commands might be found useful:

- command to show currently used Docker images in Kubernetes Listing 5,
- command to delete images from remote Artifactory Listing 6,
- command to show currently stored Docker images in remote Artifactory [75] Listing 7,
- Script to remove stored Docker images in remote Artifactory based on ids stored in txt fileListing 8.

Listing 5: Command to show currently used Docker images in Kubernetes.

```
kubectl get pods --all-namespaces -o
↪  jsonpath="{.items[*].spec.containers[*].image}" |tr -s
↪  '[[:space:]]' '\n' |sort |uniq -c | tr -s " " | cut -f3 -d" "
```

Listing 6: Command to delete images from remote Artifactory.

```
curl -u 'user:pass' -X DELETE
↪  "https://artifactory-address.net/artifactory/
↪  folder-name/image-id"
```

Listing 7: Command to show currently stored Docker images in remote Artifactory.

```
curl -u 'user:pass'  "https://artifactory-address.net/
↪  artifactory/api/storage/ml-ci/"
```

Listing 8: Script to remove stored Docker images in remote Artifactory based on ids stored in txt file.

```python
import requests

with open("images_to_remove_ids.txt") as f_images_to_remove_ids:
    images = f_images_to_remove_ids.readlines()

images = [image.strip() for image in images]
for image_id in images:
    if image_id:
        requests.delete(
            "https://artifactory-address.net/",
            f"artifactory/folder-name/{image_id}",
            auth=("user", "pass"),
        )
```

7.7 Installation of GitLab runner

GitLab [2] is a solution to managing Git [33] repositories. It enables Continuous Integration and Continuous Delivery (CI/CD). This section is based on instructions in [92]. If there is a need to create standalone GitLab runner then extract token obtained during registration from `/etc/gitlab-runner/config.toml`. Standard registration from a standalone machine can be done with the command shown in Listing 9. Registration tokens can be gathered from Settings tab CI/CD section. After registration, the token should be filled into the ConfigMap resource which general concepts are described in [5]), whereas the example of required one is shown in Listing 11. A namespace called gitlab-runner has to be created. This step is required to the apply configuration of Service Account described in [7] and shown in Listing 10. In the final step, file shown in Listing 12 should be used to create the Deployment [6].

Listing 9: Command to register runner.

```
gitlab-runner register --name gitlab-runner-data-selector --url
↪  https://gitlab-address.com/ --registration-token ***
```

Listing 10: Creating Service Account.

```yaml
apiVersion: v1
kind: ServiceAccount
metadata:
  name: gitlab-admin
  namespace: gitlab-runner
---
kind: Role
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  namespace: gitlab-runner
  name: gitlab-admin
rules:
  - apiGroups: [""]
    resources: ["*"]
    verbs: ["*"]

---
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: gitlab-admin
  namespace: gitlab-runner
subjects:
  - kind: ServiceAccount
    name: gitlab-admin
    namespace: gitlab-runner
roleRef:
  kind: Role
  name: gitlab-admin
  apiGroup: rbac.authorization.k8s.io

```

Listing 11: ConfigMap of GitLab Runner Settings contains among other standard TOML config content.

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: gitlab-runner-config
  namespace: gitlab-runner
data:
  config.toml: |-
    concurrent = 4
    [[runners]]
      environment = ["DOCKER_AUTH_CONFIG={\"auths\": {\"artifactory-address.net\":{\"auth\"\"\"***\"}}}", "FUNCTIONAL_ACCOUNT_LOGIN=***","FUNCTIONAL_ACCOUNT_PASSWORD=***"]
      pre_build_script = "mkdir ~/.docker -p && echo $DOCKER_AUTH_CONFIG > ~/.docker/config.json && mkdir -p /etc/functional_account_0 && echo ${FUNCTIONAL_ACCOUNT_LOGIN} > /etc/functional_account_0/functional_account_0_login && echo ${FUNCTIONAL_ACCOUNT_PASSWORD} > /etc/functional_account_0/functional_account_0_password"
      name = "Kubernetes Runner"
      url = "https://gitlab-address.com/"
      token = "***"
      executor = "kubernetes"
      [runners.kubernetes]
        namespace = "gitlab-runner"
        poll_timeout = 600
        cpu_request = "1"
        service_cpu_request = "200m"
        privileged = true
        [[runners.kubernetes.pod_spec]]
          name = "val1 node"
          patch = '''
            [{ "op": "add", "path": "/nodeSelector", "value": { "environment": "production" } },{ "op": "add", "path": "/tolerations", "value": { "key": "environment", "operator": "Equal", "value": "production", "effect": "NoSchedule" } }]
          '''
          patch_type = "json"

```

Listing 12: GitLab runner deployment.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: gitlab-runner
  namespace: gitlab-runner
spec:
  replicas: 1
  selector:
    matchLabels:
      name: gitlab-runner
  template:
    metadata:
      labels:
```

```yaml
        name: gitlab-runner
    spec:
      tolerations:
      - key: "environment"
        operator: "Equal"
        value: "research"
        effect: "NoSchedule"
      nodeSelector:
        environment: research
      serviceAccountName: gitlab-admin
      containers:
        - args:
          - run
          image: gitlab/gitlab-runner:latest
          imagePullPolicy: Always
          name: gitlab-runner
          resources:
            requests:
              cpu: "100m"
            limits:
              cpu: "100m"
          volumeMounts:
            - name: config
              mountPath: /etc/gitlab-runner/config.toml
              readOnly: true
              subPath: config.toml
      volumes:
        - name: config
          configMap:
            name: gitlab-runner-config
      restartPolicy: Always
```

7.8 Installation of Renovate Bot

Renovate Bot [11, 12] is a solution which automatically indicates the recent updates for packages available. It creates new merge requests with proposals for changes. It can be installed by using a package manager for Kubernetes called Helm [4]. Default config is in public repository [13]. There is a need to change part of the yaml configuration placing the appropriate token and repositories. A token is created as described in manual [3]. Example of that part is shown in Listing 13.

Listing 13: Renovate Bot part of config which has to be updated.

```yaml
config: |
    {
      "platform": "gitlab",
      "endpoint": "https://gitlab-address.com/api/v4",
      "token": "glpat-v-z*****z",
      "autodiscover": "true",
      "dryRun": false,
      "printConfig": true,
      "repositories": ["filtering_service","project_template"],
      "pip_requirements": {
        "fileMatch": ["requirements/base.txt"]
      }
    }

```

Listing 14: Commands required to install Renovate Bot.

```
kubectl create ns renovate-bot-helm
helm repo add renovate https://docs.renovatebot.com/helm-charts
helm upgrade --install --namespace renovate-bot-helm rosie
 ↪  renovate/renovate -f values.yml

```

CHAPTER 8

Conclusions and future works

This paper shows the area of use of machine learning in applications related to bug reports in software development. The main research objectives of the work were related to assignment of software bug reports to specific development groups. This was done at different levels of organization. In addition, research about assigning other security-related classes was conducted. For both above cases, the possibilities of using methods related to the explainable artificial intelligence were analyzed. The potential application of a multimodal neural network to solve the problem of assigning error reports in software to groups responsible for their correction or analysis was also investigated. During the evaluation of the possibility of improving industrial implementations, some of the results of the evaluation were compared with the results of human prediction.

One of the key methods which have been examined in this work was the possibility of application of multimodal neural network for software bug report assignment. The study including numerical simulations showed possible benefits of such solution in better accuracy comparing to reference methods. The inputs which were used come from title, description and extracted data from log archives represented with the use of TF-IDF and categorical data.

The main thesis stated in the work is:

There exists a method for automated assignment of a software bug report to appropriate development group, responsible for resolving the bug, which outperforms well-known methods for bug report assignment.

Thesis statement is confirmed in this dissertation. In the first material [38], the results are improved by extending utilized bug report data of features about information related to product type what gave a positive impact on accuracy. Paper [36] (Chapter 6 which is planned to be published) presents application of multimodal neural network for software bug report assignment. Research in this material utilized data of software bug report such as title, description, categorical fields from software bug report and data extracted from respective log archives. The use of multimodal neural network outperforms referenced state-of-the-art solutions.

Time dependencies presented in [40] introduce important factor for evaluation, especially when taken into consideration duplicates and duration of solving issue related to bug report in the context of continous software development. This fact may have an impact on results, because the model used for production purposes cannot be trained with the recently submitted (or not yet resolved) bug reports. This restriction is not obeyed in the case of utilizing random split in building sets for evaluation.

Different research in this dissertation focused among others on potential application of explainable artificial intelligence, software bug report classification and labelling. In the course of conducting the research, peculiar discoveries were reported in the company's internal process regarding Intellectual Properties Right. As a result, three internal patent-like applications were prepared. Processing one of the documents, the company purchased a prior art search. The patentability search report did not identify references in worldwide patent and non-patent literature, confirming a decent level of novelty. Eventually, NOKIA decided to make a publication and did not file the patent request. That content has been published in IAPGOS [39]. Other selected results which were also reported in the Intellectual Properties Right process were published in Scientific Reports of Nature [40].

Based on the results of the analysis, the innovative software bug report assignment method was implemented at NOKIA. Selected parameters from research on changes in preprocessing and vectorization of text were used and implemented in company systems. These results were presented at *the 12th International Conference on Computer Recognition Systems* and printed as a chapter in a peer-reviewed international monograph of international importance published by Springer [38].

In addition, a solution with the selection of confidence thresholds when combining predictions was implemented. Detailed results related to that novel method for software bug assignment were published in [37]. This implementation took place gradually. The development group responsible for analysis of software bug reports within the department was supported by a tool. The system entered suggestions to which division the member of above-mentioned development group should transfer bug report for further processing. At a later stage, for selected reports, a decision was made to transfer individual error reports automatically. Later the quantity of automatically transferred cases increased. Subsequently, from the managerial side a decision was made to completely omit the above-mentioned group from the chain of analysis of bug reports and transfer software bug reports directly to divisions. As a result, cases which are transferred with the use of the tool into department areautomatically assigned directly to divisions.

All research questions, Section 1.5, have been discussed and answered directly or indirectly inside articles [36, 37, 38, 39, 40]. These papers discuss topics related to impact of stemming and lemmatization on bug assignment accuracy, comparison of results of machine learning predictions versus humans predictions, benefits of the proposed novel method of software bug report assignment. Potential application of XAI and multimodal neural network were discussed. A few key findings related to these two approaches were also shown.

This research is targeted for assignment a software bug report to proper group which should be involved in resolving problem. The list of accomplishments, extending knowledge in software bug report assignment area, includes:

- compared possible results which might be achieved with usage of title, description and product versus results based on features extracted from snapshot(s);
- discussed limitations of state of the art evaluation methods;
- conducted exploratory research about comparison simulation results against human predictions;
- conducted conclusive research about potential benefits of application of automated assignment of bug at current/improved state of the art;
- analyzed application of XAI methods in software bug assignment or labeling;
- examined possibility of application of multimodal neural network in software bug report assignment problems.

Selected implementations introduced in the company during PhD studies are among others listed below and in Appendix C. Currently, there is ongoing work related to implementing multimodal neural network into the company system. This includes hyperparameter optimization, feature selection, and handling missing data. Nevertheless, the first version of the system with multimodal neural network has already been introduced. Moreover, many smaller improvements related to handling software bug report processes were also introduced, including both those related to machine learning and not. One of the solutions was upgraded using part of parameters related to preprocessing and vectorization further described in Chapter 2. What is more, there has also been introduced a solution related to transferring software bug reports between groups in case it meets specific conditions among others related to confidence level thresholds and appropriate log content presence. Work is currently underway to implement solutions related to explainable artificial intelligence and multimodal neural network in software bug report solutions within the company.

The software bug report assignment enhanced with the use of machine learning techniques is discussed in literature and some results are publicly available. Taking into account that this topic is in interest of mainly big software companies and thus, the research work results in this area are rather modest. Smaller companies are less interested in such approaches as there is no need for highly complex methods of automatic software bug report assignment. Smaller projects require fewer developers and thus the task of software bug report assignment is usually much simpler. Eventually the software bug report assignment is a research niche.

As a future work it is planned to extend research related to application of multimodal neural networks of another type of inputs which will be gathered from logs archives, but with the use of different representations of data.

REFERENCES

[1] Iso/iec/ieee international standard - software engineering - software life cycle processes - maintenance. *ISO/IEC/IEEE 14764:2022(E)*, pages 1–46, 2022.

[2] Gitlab. `https://about.gitlab.com/`, 2023. [Online; accessed 15-Jun-2023].

[3] Gitlab docs - personal access tokens. `https://docs.gitlab.com/ee/user/profile/personal_access_tokens.html`, 2023. [Online; accessed 30-Aug-2023].

[4] Helm docs. `https://helm.sh/docs/`, 2023. [Online; accessed 30-Aug-2023].

[5] Kubernetes - configmaps. `https://kubernetes.io/docs/concepts/configuration/configmap`, 2023. [Online; accessed 15-Jun-2023].

[6] Kubernetes - deployments. `https://kubernetes.io/docs/concepts/workloads/controllers/deployment/`, 2023. [Online; accessed 15-Jun-2023].

[7] Kubernetes - service accounts. `https://kubernetes.io/docs/concepts/security/service-accounts/`, 2023. [Online; accessed 15-Jun-2023].

[8] Kubernetes manual. `https://kubernetes.io/docs/home/`, 2023. [Online; accessed 02-Mar-2023].

[9] Performing a rolling update, May 2023. "https://kubernetes.io/docs/tutorials/kubernetes-basics/update/update-intro/".

[10] Powerbi. `https://www.microsoft.com/en-us/power-platform/products/power-bi/`, 2023. [Online; accessed 19-Mar-2024].

[11] Renovate docs. `https://docs.renovatebot.com/`, 2023. [Online; accessed 30-Aug-2023].

[12] Renovate docs. `https://github.com/renovatebot/helm-charts/tree/main`, 2023. [Online; accessed 30-Aug-2023].

[13] Renovate docs. `https://github.com/renovatebot/helm-charts/blob/main/charts/renovate/values.yaml`, 2023. [Online; accessed 30-Aug-2023].

[14] S. N. Ahsan, J. Ferzund, and F. Wotawa. Automatic software bug triage system

(bts) based on latent semantic indexing and support vector machine. In *2009 Fourth International Conference on Software Engineering Advances*, pages 216–221, 2009.

[15] I. Alazzam, A. Aleroud, Z. Al Latifah, and G. Karabatis. Automatic bug triage in software systems using graph neighborhood relations for feature augmentation. *IEEE Transactions on Computational Social Systems*, 7(5):1288–1303, 2020.

[16] R. Aleithan. Explainable just-in-time bug prediction: Are we there yet? In *2021 IEEE/ACM 43rd International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, pages 129–131, 2021.

[17] J. Ali, M. Adnan, T. R. Gadekallu, R. H. Jhaveri, and B.-H. Roh. A qos-aware software defined mobility architecture for named data networking. In *2022 IEEE Globecom Workshops (GC Wkshps)*, pages 444–449, 2022.

[18] G. An, M. Akiba, K. Omodaka, T. Nakazawa, and H. Yokota. Hierarchical deep learning models using transfer learning for disease detection and classification based on small number of medical images. *Scientific Reports*, 11(1):4250, Mar 2021.

[19] Anjali, D. Mohan, and N. Sardana. Visheshagya: Time based expertise model for bug report assignment. In *2016 Ninth International Conference on Contemporary Computing (IC3)*, pages 1–6, 2016.

[20] O. A. Antwi, A. O. Owusu, J. W. Nanjo, G. B. Gidisu, D. Sackey, and H. Mohammed. Analysis of collocated base transceiver stations and associated risks in erecting base stations. In *2021 International Conference on Computing, Computational Modelling and Applications (ICCMA)*, pages 92–97, 2021.

[21] F. Ariza-Lopez, J. Rodriguez-Avi, and M. Alba-Fernandez. Complete control of an observed confusion matrix. In *IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium*, pages 1222–1225, 2018.

[22] J. Banda, R. Angryk, and P. Martens. Steps toward a large-scale solar image data analysis to differentiate solar phenomena. *Solar Physics*, 288, 05 2013.

[23] A. Barredo Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. Garcia, S. Gil-Lopez, D. Molina, R. Benjamins, R. Chatila, and F. Herrera. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. volume 58, pages 82–115, 2020.

[24] A. G. Barto. Adaptive real-time dynamic programming. In C. Sammut and G. I. Webb, editors, *Encyclopedia of Machine Learning*, pages 19–22. Springer, 2010.

[25] G. Bebis and M. Georgiopoulos. Feed-forward neural networks. *IEEE Potentials*, 13(4):27–31, 1994.

[26] D. Behl, S. Handa, and A. Arora. A bug mining tool to identify and analyze security bugs using naive bayes and tf-idf. In *2014 International Conference on*

Reliability Optimization and Information Technology (ICROIT), pages 294–299, 2014.

[27] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[28] H. B. Borges, C. N. Silla, and J. C. Nievola. An evaluation of global-model hierarchical classification algorithms for hierarchical classification problems with single path of labels. *Computers & Mathematics with Applications*, 66(10):1991–2002, 2013. ICNC-FSKD 2012.

[29] J. Brownlee. Machine learning mastery: A gentle introduction to k-fold cross-validation. `https://machinelearningmastery.com/k-fold-cross-validation/`, 10 2022.

[30] F. Calefato, F. Lanubile, F. Maiorano, and N. Novielli. Sentiment polarity detection for software development. pages 128–128, 05 2018.

[31] A. Carlevaro, M. Lenatti, A. Paglialonga, and M. Mongelli. Counterfactual building and evaluation via explainable support vector data description. *IEEE Access*, 10:60849–60861, 2022.

[32] A. Carlevaro and M. Mongelli. A new svdd approach to reliable and explainable ai. *IEEE Intelligent Systems*, 37(2):55–68, 2022.

[33] S. Chacon and B. Straub. *Pro Git*. Apress, 2014.

[34] X. Chang and W. Skarbek. Multi-modal residual perceptron network for audio–video emotion recognition. *Sensors*, 21(16), 2021.

[35] Q. Cheng, C. Zhang, and X. Shen. Estimation of energy and time usage in 3d printing with multimodal neural network. In *2022 4th International Conference on Frontiers Technology of Information and Computer (ICFTIC)*, pages 900–903, 2022.

[36] L. Chmielowski, P. Konstantynov, R. Luczak, M. Kucharzak, and R. Burduk. Application of multimodal neural networks in solving problem of labeling bug reports. Unpublished material.

[37] L. Chmielowski, P. Konstantynov, R. Luczak, M. Kucharzak, and R. Burduk. A novel method for software bug report assignment. Reliability Theory and Applications.

[38] L. Chmielowski and M. Kucharzak. Impact of software bug report preprocessing and vectorization on bug assignment accuracy. In M. Choraś, R. S. Choraś, M. Kurzyński, P. Trajdos, J. Pejaś, and T. Hyla, editors, *Progress in Image Processing, Pattern Recognition and Communication Systems*, pages 153–162, Cham, 2022. Springer International Publishing.

[39] L. Chmielowski, M. Kucharzak, and R. Burduk. Application of explainable artificial intelligence in software bug classification. *Informatyka, Automatyka, Po-*

miary w Gospodarce i Ochronie Srodowiska, 13(1):14–17, Mar. 2023.

[40] L. Chmielowski, M. Kucharzak, and R. Burduk. Novel method of building train and test sets for evaluation of machine learning models related to software bugs assignment. *Scientific Reports*, 13, 12 2023.

[41] C. A. Choquette-Choo, D. Sheldon, J. Proppe, J. Alphonso-Gibbs, and H. Gupta. A multi-label, dual-output deep neural network for automated bug triaging. In *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, pages 937–944, 2019.

[42] K. Constantinides, O. Mutlu, T. Austin, and V. Bertacco. A flexible software-based framework for online detection of hardware defects. *IEEE Transactions on Computers*, 58(8):1063–1079, 2009.

[43] D. Cubranic and G. Murphy. Automatic bug triage using text categorization. In *SEKE*, 2004.

[44] J. A. de Bruijn, H. de Moel, A. H. Weerts, M. C. de Ruiter, E. Basar, D. Eilander, and J. C. Aerts. Improving the classification of flood tweets with contextual hydrological information in a multimodal neural network. *Computers & Geosciences*, 140:104485, 2020.

[45] M. Dutta. BEFORE AND AFTER OF DevOps: A PEEK INTO AGILE DevOps, Nov 2019. https://medium.com/mainakdutta76/before-and-after-of-devops-a-peek-into-agile-devops-3600c26129ac.

[46] B. El Khalyly, A. Belangour, M. Banane, and A. Erraissi. A new metamodel approach of ci/cd applied to internet of things ecosystem. In *2020 IEEE 2nd International Conference on Electronics, Control, Optimization and Computer Science (ICECOCS)*, pages 1–6, 2020.

[47] P. Flach. Performance evaluation in machine learning: The good, the bad, the ugly, and the way forward. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):9808–9814, Jul. 2019.

[48] J. Fürnkranz. *Decision Tree*, pages 263–267. Springer US, Boston, MA, 2010.

[49] J. L. Garcia-Balboa, M. V. Alba-Fernandez, F. J. Ariza-López, and J. Rodriguez-Avi. Homogeneity test for confusion matrices: A method and an example. In *IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium*, pages 1203–1205, 2018.

[50] N. Geographic. World's first computer bug, Oct 2020. "https://www.nationalgeographic.org/thisday/sep9/worlds-first-computer-bug/".

[51] R. Gerhards. The Syslog Protocol. RFC 5424 (Proposed Standard), March 2009.

[52] Y. Goldberg and O. Levy. word2vec explained: deriving mikolov et als negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*, 2014.

[53] 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)*, pages 344–355, 2018.

[54] S. S. Group. 10 historical software bugs with extreme consequences, Jan 2017. "https://safebytes.com/10-historical-software-bugs-extreme-consequences/".

[55] S. Gujral, G. Sharma, S. Sharma, and Diksha. Classifying bug severity using dictionary based approach. In *2015 International Conference on Futuristic Trends on Computational Analysis and Knowledge Management (ABLAZE)*, pages 599–602, 2015.

[56] J. Han, M. Kamber, and J. Pei. 2 - getting to know your data. In J. Han, M. Kamber, and J. Pei, editors, *Data Mining (Third Edition)*, The Morgan Kaufmann Series in Data Management Systems, pages 39–82. Morgan Kaufmann, Boston, third edition edition, 2012.

[57] M. Heydarian, T. E. Doyle, and R. Samavi. Mlcm: Multi-label confusion matrix. *IEEE Access*, 10:19083–19095, 2022.

[58] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[59] I. S. T. Institute. Software testing levels, Jan 2021. https://www.test-institute.org/Software_Testing_Levels.php.

[60] ISO/IEC/IEEE 29119-3:2021 Software and systems engineering — Software testing — Part 3: Test documentation. Standard, International Organization for Standardization, Oct. 2021.

[61] H. Jabeen. Stemming and Lemmatization in Python, Oct 2018. https://www.datacamp.com/community/tutorials/stemming-lemmatization-python.

[62] L. Jonsson. *Machine Learning-Based Bug Handling in Large-Scale Software Development*. PhD Thesis, Linköping Studies in Science and Technology, 2018.

[63] 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)*, pages 406–409, 2019.

[64] Z. Karimi. Confusion matrix. `https://www.researchgate.net/publication/355096788`, 10 2021. [Online; accessed 13-October-2022].

[65] C. Khanan, W. Luewichana, K. Pruktharathikoon, J. Jiarpakdee, C. Tantithamthavorn, M. Choetkiertikul, C. Ragkhitwetsagul, and T. Sunetnanta. Jitbot: An explainable just-in-time defect prediction bot. In *2020 35th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 1336–1339, 2020.

[66] R. Kiros, R. Salakhutdinov, and R. Zemel. Multimodal neural language models.

In E. P. Xing and T. Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 595–603, Bejing, China, 22–24 Jun 2014. PMLR.

[67] T. Kramer, F. Proctor, and E. Messina. The nist rs274ngc interpreter - version 3, 2000-08-01 2000.

[68] Docker manual. `https://docs.docker.com/`, 2023. [Online; accessed 02-Mar-2023].

[69] B. Kucuk and E. Tuzun. Characterizing duplicate bugs: An empirical analysis. In *2021 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pages 661–668, 2021.

[70] A. Lamkanfi and S. Demeyer. Predicting reassignments of bug reports - an exploratory investigation. In *2013 17th European Conference on Software Maintenance and Reengineering*, pages 327–330, 2013.

[71] A. Lamkanfi, J. Pérez, and S. Demeyer. The eclipse and mozilla defect tracking dataset: A genuine dataset for mining bug information. In *2013 10th Working Conference on Mining Software Repositories (MSR)*, pages 203–206, 2013.

[72] T. Landauer, P. Foltz, and D. Laham. An introduction to latent semantic analysis. *Discourse Processes*, 25:259–284, 01 1998.

[73] B. T. Lee, L. Masinter, and M. Mccahill. RFC 1738: Uniform resource locator (URL). http://www.ietf.org/rfc/rfc1738.txt, 1994.

[74] M. Lee, Y. Zhu, K. Srinivasan, P. Shah, S. Savarese, L. Fei-Fei, A. Garg, and J. Bohg. Making sense of vision and touch: Self-supervised learning of multimodal representations for contact-rich tasks. pages 8943–8950, 05 2019.

[75] J. Light, P. Pfeiffer, and B. Bennett. An evaluation of continuous integration and delivery frameworks for classroom use. In *Proceedings of the 2021 ACM Southeast Conference*, ACM SE '21, page 204–208, New York, NY, USA, 2021. Association for Computing Machinery.

[76] H. Lin, H. Sheng-zong, W. You-liang, L. Xiao-si, and J. Qi-zheng. Study on electrostatic discharge damage and defect damage failure analysis technology for semiconductor devices. In *2018 19th International Conference on Electronic Packaging Technology (ICEPT)*, pages 1246–1249, 2018.

[77] T. D. Lingayat. Prediction of electrostatic discharge soft failure issue in case of a six layer pcb of a tablet using siwave tool. In *2016 IEEE International Conference on Recent Trends in Electronics, Information and Communication Technology (RTEICT)*, pages 1361–1366, 2016.

[78] K. Liu, H. Beng Kuan Tan, and H. Zhang. Has this bug been reported? In *2013 20th Working Conference on Reverse Engineering (WCRE)*, pages 82–91, 2013.

[79] V. M. P. M. Abavisani. Deep multimodal subspace clustering networks. 2018.

[80] e. a. M. Castelluccio. bugbug, Apr 2019. https://github.com/mozilla/bugbug.

[81] S. L. M. Castelluccio. Teaching machines to triage firefox bugs, Apr 2019. https://hacks.mozilla.org/2019/04/teaching-machines-to-triage-firefox-bugs/.

[82] H. Mahfoodh and M. Hammad. Word2vec duplicate bug records identification prediction using tensorflow. In *2020 International Conference on Innovation and Intelligence for Informatics, Computing and Technologies (3ICT)*, pages 1–6, 2020.

[83] H. Mahfoodh and Q. Obediat. Software risk estimation through bug reports analysis and bug-fix time predictions. In *2020 International Conference on Innovation and Intelligence for Informatics, Computing and Technologies (3ICT)*, pages 1–6, 2020.

[84] S. Matzka. Explainable artificial intelligence for predictive maintenance applications. In *2020 Third International Conference on Artificial Intelligence for Industries (AI4I)*, pages 69–74, 2020.

[85] D. Merkel. Docker: lightweight linux containers for consistent development and deployment. *Linux journal*, 2014(239):2, 2014.

[86] T. S. Mian. Automation of bug-report allocation to developer using a deep learning algorithm. In *2021 International Congress of Advanced Technology and Engineering (ICOTEN)*, pages 1–7, 2021.

[87] U. Michelucci. An introduction to autoencoders. *CoRR*, abs/2201.03898, 2022.

[88] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *Proceedings of Workshop at ICLR*, 2013, 01 2013.

[89] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space, 2013.

[90] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26, 2013.

[91] M. Monperrus. Explainable software bot contributions: Case study of automated bug fixes. In *2019 IEEE/ACM 1st International Workshop on Bots in Software Engineering (BotSE)*, pages 12–15, 2019.

[92] B. Murugan. Configure gitlab runner on kubernetes. `https://www.linkedin.com/pulse/configure-gitlab-runner-kubernetes-bala-murugan`, 2023. [Online; accessed 15-Jun-2023].

[93] K. Naik and P. Tripathy. Software testing and quality assurance: Theory and practice. pages 601–616, 02 2008.

[94] V. Nath, D. Sheldon, and J. Alphonso-Gibbs. Principal component analysis and

entropy-based selection for the improvement of bug triage. In *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 541–546, 2021.

[95] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Ng. Multimodal deep learning. pages 689–696, 01 2011.

[96] K. O'Shea and R. Nash. An introduction to convolutional neural networks. *CoRR*, abs/1511.08458, 2015.

[97] X. Ouyang, P. Zhou, C. H. Li, and L. Liu. Sentiment analysis using convolutional neural network. In *2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing*, pages 2359–2364, 2015.

[98] G. O'Regan. *Introduction to Software Quality*. Undergraduate Topics in Computer Science. Springer International Publishing, 2014.

[99] 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*, 12:2825–2830, 2011.

[100] 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, confusionmatrixdisplay. *Journal of Machine Learning Research*, 12:2825–2830, 2022.

[101] 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, tuning the hyper-parameters of an estimator. *Journal of Machine Learning Research*, 12:2825–2830, 2022.

[102] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[103] M. Pereira, A. Kumar, and S. Cristiansen. Identifying security bug reports based solely on report titles and noisy data. In *2019 IEEE International Conference on Smart Computing (SMARTCOMP)*, pages 39–44, 2019.

[104] U. N. H. C. Photograph. The first "computer bug", Oct 2020. "`https://web.archive.org/web/20150112215748/http://www.history.navy.mil/photos/images/h96000/h96566kc.htm`".

[105] S. Raschka. 5x2cv paired ttest, Jan 2021. `https://rasbt.github.io/mlxtend/user_guide/evaluate/paired_ttest_5x2cv`.

[106] P. Refaeilzadeh, L. Tang, and H. Liu. *Cross-Validation*. 2009.

[107] R. K. Roul, J. K. Sahoo, and K. Arora. Modified tf-idf term weighting strategies for text categorization. In *2017 14th IEEE India Council International Conference (INDICON)*, pages 1–6, 2017.

[108] M. Rungta, P. P. Sherki, M. P. Dhaliwal, H. Tiwari, and V. Vala. Two-phase multimodal neural network for app categorization using apk resources. In *2020 IEEE 14th International Conference on Semantic Computing (ICSC)*, pages 162–165, 2020.

[109] M. M. Saad, M. T. R. Khan, G. Srivastava, R. H. Jhaveri, M. Islam, and D. Kim. Cooperative vehicular networks: An optimal and machine learning approach. *Computers and Electrical Engineering*, 103:108348, 2022.

[110] M. Salahat, R. A. Said, K. Hamid, U. Haseeb, E. Abdel Maguid Abdel Ghani, A. Abualkishik, M. W. Iqbal, and M. Inairat. Software testing issues improvement in quality assurance. In *2023 International Conference on Business Analytics for Technology and Security (ICBATS)*, pages 1–6, 2023.

[111] C. Sammut and G. I. Webb, editors. *Leave-One-Out Cross-Validation*, pages 600–601. Springer US, Boston, MA, 2010.

[112] A. Sarkar, P. C. Rigby, and B. Bartalos. Improving bug triaging with high confidence predictions at ericsson. In *2019 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 81–91, 2019.

[113] M. C. Schneider. *Mosfet Modeling for Circuit Analysis And Design*. World Scientific Publishing Co., Inc., USA, 2007.

[114] K. Singh and A. Raut. Feature selection for anomaly based intrusion detection using rough set theory. 04 2014.

[115] M. Sokolova and G. Lapalme. A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4):427–437, 2009.

[116] H. Tahir, S. U. R. Khan, and S. S. Ali. Lcbpa: An enhanced deep neural network-oriented bug prioritization and assignment technique using content-based filtering. *IEEE Access*, 9:92798–92814, 2021.

[117] A. Tsuruda, Y. Manabe, and M. Aritsugi. Can we detect bug report duplication with unfinished bug reports? In *2015 Asia-Pacific Software Engineering Conference (APSEC)*, pages 151–158, 2015.

[118] Q. Umer, H. Liu, and I. Illahi. Cnn-based automatic prioritization of bug reports. *IEEE Transactions on Reliability*, 69(4):1341–1354, 2020.

[119] L. A. Vayghan, M. A. Saied, M. Toeroe, and F. Khendek. Kubernetes as an

Availability Manager for Microservice Applications. *arXiv/CoRR*, 2019.

[120] G. Vilone and L. Longo. Explainable artificial intelligence: a systematic review. volume abs/2006.00093, 2020.

[121] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[122] G. I. Webb. *Naïve Bayes*, pages 713–714. Springer US, Boston, MA, 2010.

[123] Wikipedia contributors. Confusion matrix — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Confusion_matrix&oldid=1107701525`, 2022. [Online; accessed 13-October-2022].

[124] Wikipedia contributors. Hardware bug — Wikipedia, the free encyclopedia, 2022. [Online; accessed 14-August-2022].

[125] J. Wu. Multi platform public opinion risk monitoring and early warning algorithm based on multimodal neural network. In *2022 International Conference on Knowledge Engineering and Communication Systems (ICKES)*, pages 1–5, 2022.

[126] G. Xiao, X. Du, Y. Sui, and T. Yue. Hindbr: Heterogeneous information network based duplicate bug report prediction. In *2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE)*, pages 195–206, 2020.

[127] Y. Xu, C. Liu, Y. Li, Q. Xie, and H.-D. Choi. A method of component prediction for crash bug reports using component-based features and machine learning. In *2023 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pages 773–777, 2023.

[128] Q. Yang, G. Wu, Y. Li, R. Li, X. Gu, H. Deng, and J. Wu. Amnn: Attention-based multimodal neural network model for hashtag recommendation. *IEEE Transactions on Computational Social Systems*, 7(3):768–779, 2020.

[129] P. D. Yasen Jiao. Performance measures in evaluating machine learning based bioinformatics predictors for classifications. *Quantitative Biology*, 4(4):320, 2016.

[130] S. F. A. Zaidi, F. M. Awan, M. Lee, H. Woo, and C.-G. Lee. Applying convolutional neural networks with different word representation techniques to recommend bug fixers. *IEEE Access*, 8:213729–213747, 2020.

[131] S. F. A. Zaidi and C. G. Lee. Learning graph representation of bug reports to triage bugs using graph convolution network. In *2021 International Conference on Information Networking (ICOIN)*, pages 504–507, 2021.

[132] S. F. A. Zaidi and C. G. Lee. One-class classification based bug triage system to assign a newly added developer. In *2021 International Conference on Information Networking (ICOIN)*, pages 738–741, 2021.

[133] S. F. A. Zaidi, H. Woo, and C.-G. Lee. A graph convolution network-based bug triage system to learn heterogeneous graph representation of bug reports. *IEEE Access*, 10:20677–20689, 2022.

[134] S. Zhang, J. Zhai, B. Xie, Y. Zhan, and X. Wang. Multimodal representation learning: advantages, trends and challenges. 2019.

[135] W. Zhang. Efficient bug triage for industrial environments. In *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 727–735, 2020.

[136] C. Zheng, L. Pan, and P. Wu. Multimodal deep network embedding with integrated structure and attribute information. *IEEE Transactions on Neural Networks and Learning Systems*, 31(5):1437–1449, 2020.

[137] C. Zhou, B. Li, X. Sun, and H. Guo. Recognizing software bug-specific named entity in software bug repository. In *2018 IEEE/ACM 26th International Conference on Program Comprehension (ICPC)*, pages 108–10811, 2018.

Appendices

Appendix A

Template of bug report

Listing A.1: Template of bug report

```
[1. Detail Test Steps:]

[2. Expected Result:]

[3. Actual Result:]

[4. Tester analysis:]

[5. Internal tool information:]

[6. Log(s) file name containing a fault: (clear indication (exact file name) and
    timestamp where fault can be found in attached logs):]

[7. Test-Line Reference/used HW/configuration/tools/SW version:]

[8. Used Flags: (list here used R  &  D flags):]

[9. Test Scenario History of Execution: (what was changed since it was tested
    successfully for the last time):]

Was Test Scenario passing before? ( YES | NO | New scenario )

What was the last SW version Test Scenario was passing? ( SW load | New scenario )

Were there any differences between test-lines since last time Test Scenario was
    passing?  ( YES, explanation | NO | New test-line )

Were there any changes in Test Scenario since last run it passed? ( YES, explanation |
     NO | New scenario )

[10. Test Case Reference: (QC, RP or UTE link):]
```

Appendix B

User/Developer guide for accessing machine learning based bug assignment predictions (Anonymized version)

author:
Lukasz Chmielowski

partially reviewed by:
Pavlo Konstantynov
Ryszard Luczak

B.1 Introduction

B.1.1 General purpose

Machine learning predictions of bug assignment are being used in software development. This user guide is related to selected internal services with following endpoints (section B.2). One of the ends of communication channel is called API Endpoint in REST API Architecture of applications [1]. To access each of those endpoints (URLs) where are serving models, we have to have access to NOKIA intranet. The general purpose of each of the following is to predict group(s) which should be involved in investigation or solving issues. Machine learning predictions are made with usage of a part of bug report based on internal standardized formats described with details in internal documents which presents its golden standard . Additionally, information about harmonized way of creating title of issue and description can be found in internal documents about NOKIA Fault Management process. Below described services are created to deliver machine learning predictions based mainly on natural language description extracted from title and description and categorical fields. They also utilize log content placed in description. Standardized API (Application Programming Interface) is described in section B.2.3. That part is sometimes known as a contract between parties which create software. Here we have defined the way of communication between micro-services serving machine learning predictions and part of application(s) which utilize them.

There are also available predictions currently based on only log content extracted from snapshots. They are returned with usage of interface from log analyzers. Each of them is a plugin written mainly based on experts' domain knowledge to extract most important parts from the log content. Via that interfaces related to plugins developers provide suggestions about the possible next group which should be engaged in analysis. So that interface can be utilized for both machine learning predictions and suggestions based on strict rules.

B.2 Available services

B.2.1 Cross department predictions

B.2.1.1 Cross department predictions standard

This model for predictions cross departments is available under URL:

`https://cross-department-predictions`

The purpose of that service is to deliver machine learning predictions related to departments. It currently returns in response results of biggest departments:

- A1,
- A2,
- A3,
- A4,
- A5,
- A6.

Designed output is following (listing 15):

Listing 15: Universal raw response

```
b'{"Group_A":0.075,"Group_B":0.351,"Group_C":0.045}\n{"Group_A":0.16⌋
↪  5,"Group_B":0.247,"Group_C":0.054}'
```

For convenience of documentation is shown formatted below (listing 16):

Listing 16: Cross department model response example

```
{
  "A1": 0.0756872861,
  "A2": 0.3516054862,
  "A3": 0.0454938939,
  "A4": 0.144986243,
  "A5": 0.2984029639,
  "A6": 0.0838241267
}
{
  "A1": 0.1659842426,
  "A2": 0.2475543452,
  "A3": 0.0542832767,
  "A4": 0.0391181373,
  "A5": 0.4385185125,
  "A6": 0.0545414858
}
```

Pandas view of the data shown above is listed below (Listing 17):

Listing 17: Cross department model response example, Pandas view

```
         A1         A2         A3         A4         A5         A6
0  0.075687  0.351605  0.045494  0.144986  0.298403  0.083824
1  0.165984  0.247554  0.054283  0.039118  0.438519  0.054541
```

B.2.1.2 A6 to A2 predictions

That API returns the decision if the case should be transferred (from A6) to A2 department. API returns empty dictionary or selected group from A2 department. The selected group is A2. This group handles cases when A6 models decide to transfer cases to A2 department. It is available under following URL:

`https://a6-to-a2`

Designed output is following and according to standard (listing 15). For convenience of documentation is shown formatted below (listing 18):

Listing 18: A6 to A2 model response example

```
{
  "A2":0.3697329487
}
```

Pandas view of the data shown above is listed below (Listing 19):

Listing 19: A6 to A2 model response example, Pandas view

```
         A2
0  0.369733
```

Note: Do not use for that API multiple rows as input, as API will return corrupted data.

B.2.1.3 A2 to A6 predictions

That API return the decision if the case should be transferred (from A2) to A6 department. API returns empty dictionary or selected group from A6 department. The selected group might be one of following from subsection B.2.2.1. This group handles cases when A6 models decides to transfer cases to A6 department. It is available under following URL:

`https://a2-to-a6`

Designed output is following and according to standard (listing 15). For convenience of documentation is shown formatted below (listing 20):

Listing 20: A2 to A6 model response example

```
{
  "A6":0.3697329487
}
```

Pandas view of the data shown above is listed below (Listing 21):

Listing 21: A2 to A6 model response example, Pandas view

```
         A6
0  0.369733
```

Note: Do not use for that API multiple rows as input, as API will return corrupted data.

B.2.2 Predictions inside department

B.2.2.1 Internal predictions standard

This service delivers predictions of competence area groups inside the department. It is available under following URL:

`https://internal-predictions`

Designed output is following and according to standard (listing 15). For convenience of documentation is shown formatted below (listing 22):

Listing 22: Internal model response example

```
{
  "A6_B1":0.0489433877,
  "A6_B2":0.3697329487,
  "A6_B3":0.1701030593,
  "A6_B4":0.0428020803,
  "A6_B5":0.0197555647,
  "A6_B6":0.0974732439,
  "A6_B7":0.1007073794,
  "A6_B8":0.0380263329,
  "A6_B9":0.1124560031
}
{
  "A6_B1":0.0857559838,
  "A6_B2":0.2583321536,
  "A6_B3":0.2105480732,
  "A6_B4":0.0409904,
  "A6_B5":0.0296723177,
  "A6_B6":0.1820736771,
  "A6_B7":0.0661168893,
  "A6_B8":0.0174926122,
  "A6_B9":0.1090178933
}
```

Pandas view of the data shown above is listed below (Listing 23):

Listing 23: Internal model response example, Pandas view

```
     A6_B1     A6_B2     A6_B3     A6_B4     A6_B5    ...
0  0.048943  0.369733  0.170103  0.042802  0.019756
1  0.085756  0.258332  0.210548  0.040990  0.029672
```

B.2.2.2 Internal predictions pilot

The purpose of this service is to return non-empty response (non-empty dictionary in response) only in case of high confidence that case would be handled inside one of groups from API from subsection B.2.2.1 and at the same time with high confidence of predictions for current department (A6). It is available under following URL:

`https://a6-internal-pilot`

Designed output is following and according to standard (listing 15). For convenience of documentation is shown formatted below (listing 24):

```
{
  "A6_B1":0.3697329487
}
```

Listing 24: Internal pilot model response example

```
b'{"Group_A":0.075,"Group_B":0.351,"Group_C":0.045}\n{"Group_A":0.16
↪  5,"Group_B":0.247,"Group_C":0.054}'
```

Pandas view of the data shown above is listed below (Listing 25):

Listing 25: Internal pilot model response example, Pandas view

```
      A6_B1
0  0.369733
```

Note: Do not use for that API multiple rows as input, as API will return corrupted data.

B.2.2.3 Internal predictions prototype 1

This prototype is available under URL:

`https://internal-predictions-prototype-1`

The output should be compatible with standard API like in listing 15. However, as it is a prototype slot for temporary testing models. These slots may have responses like any of the above in this section.

B.2.2.4 Internal predictions prototype 2

This prototype is available under URL:

`https://internal-predictions-prototype-2`

The output should be compatible with standard API like in listing 15. However as it is a prototype slot for temporary testing models. These slots may have responses like any of above in this section.

B.2.2.5 Direct hwunit mapping

This service is available under URL:

`https://direct-hwunit-mapping`

The output should be compatible with standard API like in listing 15. The purpose of this service is to directly map the case with groups for particular issues in case of presence of specific HW Units.

B.2.2.6 Direct hwunit mapping with additional department filter

This prototype is available under URL:

`https://direct-hwunit-mapping-a6-threshold`

The output should be compatible with standard API like in listing 15. The purpose of this service is to directly map the case with groups for particular issues in case of presence of specific HW Units and high probability for A6 department.

B.2.3 Example of usage

In listing 29 is placed an example of request API of multi row request. For real usage url address in line 23 should be changed for one of following described in section B.2. An example of detailed json prepared for request contains sample of full title and description is placed in Listing 26. A request of single row prediction and response for respective queries are placed in Listings 27 and 28. Numbers in response sometimes may be treated as probability, but not always. It depends on developers / machine learning engineers what will be provided via this API. In general, we can only assume that the sum of those number in a row should not be greater than 1 and the order matters like the bigger value, more confident response. Here we should remember that sometimes from models there are responses like 0.98..,0.01. It not always indicates that the response 0.98 means almost sure as part of models for manual analysis require of calculating logarithm of those value for convenient analysis. What is more users should not assume that this part and numbers will be similar for a longer period as changes in model may change the distribution of those values. The only valid assumption for long term is that the order matters, for any other please contact the author directly for more information.

Listing 26: Example of content of POST request which contains title and description fields filled

```
{"title":"[TAG1][TAG2][TAG3] example title
↪  text","description":"Sample text according to template\r\nline 2
↪  of sample text\r\nline 3 of sample text\r\nline 4 of sample
↪  text\r\n","product":"product type","softwareRelease":"release
↪  name","hwUnit":"example radio type"}
```

In line 11 of Listing 29 is a field related to id, it is not mandatory to add that field and it is not taken into consideration when the predictions are being done. Similar situation is related to key: 'healthcheck', the purpose of healthcheck field is to distinguish whether the predictions come from user requests or are made as healthcheck request, for instance, to not use them for collecting data for statistics usage.

Note: Part of API endpoints are limited to accept as input only one record at once to provide correct output or provides only selected subgroups as result. For details check respective subsections of the manual in section B.2.

Listing 27: Example of single row request to standard API

```
#!/usr/bin/env python3
import json
import logging
import pandas as pd
import requests

logging.basicConfig(level=logging.INFO)
```

```python

data = pd.DataFrame(
    data={
        "title": ["title1"],
        "description": ["desc1"],
        "softwareRelease": ["softwareRelease1"],
        "problemType": ["Software"],
        "repeatability": ["Permanent"],
        "discoveredIn": ["System Test"],
        "product": ["product1"],
        "hwUnit": ["radio1"],
    }
)
data_json = data.to_json(orient="records", lines=True)
r = requests.post(
    url=f"http://localhost",
    data=data_json,
)
logging.info(r.content)
res = pd.read_json(r.content, orient="records", lines=True)
logging.info(res)
assert r.status_code == 200
```

Listing 28: Example of response of single row request

```
INFO:root:b'{"A1":0.0827859652,"A2":0.3321372892,"A3":0.0469098809,"⌋
↪ A4":0.1578360036,"A5":0.3206158486,"A6":0.0597150126}'
INFO:root:      A1       A2        A3         A4         A5         A6
0  0.082786  0.332137  0.04691  0.157836  0.320616  0.059715
```

Listing 29: Example of multi row request to standard API

```python
#!/usr/bin/env python3
import json
import logging
import pandas as pd
import requests

logging.basicConfig(level=logging.INFO)

data = pd.DataFrame(
    data={
        "id": ["P1", "P2"],
        "title": ["title1", "title2"],
        "description": ["desc1", "desc2"],
        "release": ["release1", "release2"],
        "problemType": ["Software", "Documentation"],
        "repeatability": ["Permanent", "One Occurrence"],
        "discoveredIn": ["System Test", "Functional Test"],
            "product": ["product1", "product2"],
            "hwUnit": ["radio1", "radio2"],
            "healthcheck": [True, True],
        }
)
data_json = data.to_json(orient="records", lines=True)
r = requests.post(
    url=f"http://localhost",
    data=data_json,
)
logging.info(r.content)
res = pd.read_json(r.content, orient="records", lines=True)
logging.info(res)
assert r.status_code == 200
```

Listing 30: Example of response for multi row request

```
b'{"A1":0.0756872861,"A2":0.3516054862,"A3":0.0454938939,"A4":0.1449
↪ 86243,"A5":0.2984029639,"A6":0.0838241267}\n{"A1":0.1659842426,"
↪ A2":0.2475543452,"A3":0.0542832767,"A4":0.0391181373,"A5":0.4385
↪ 185125,"A6":0.0545414858}'
        A1        A2        A3        A4        A5        A6
0  0.075687  0.351605  0.045494  0.144986  0.298403  0.083824
1  0.165984  0.247554  0.054283  0.039118  0.438519  0.054541
```

B.3 Accessing predictions based on snapshot log content

B.3.1 Via standardized Analysis Platform API

B.3.1.1 Description

Analysis Platform is a solution which is a platform with different log analyzers. They are written as plugins. At least one of plugins is related to Radio Frequency (RF) software. Each plugin has a possibility to provide suggestions about suspected groups (groups which according to results of analysis should be involved in delivering the solution for a problem which occurred). There is also a possibility to return next to check information. Via that part is returned group which should be involved in further analysis, but the authors do not suggest that they should also deliver fix.

B.3.1.2 Example of usage

To access ML predictions based on snapshots user can utilize standard Analysis Platform API for delivering suggestions about possible suspected groups. That API in this case was used in context of machine learning based suggestions, but we should know that this is not the only purpose of its usage. Via that API suggestions about potential problems and groups responsible for investigation without any machine learning can be also sent. In below example is ID651. User should modify that part of following URL:

`https://solution.cloud/api/v2/analysissuite/bot/ID652`

, to get proper results related to particular problem report by its id. From following response (listing 31), user can get information about files attached which were detected. Each has a unique identifier made, in response as key "hash". For each file might be added separate prediction (listing 32), every of them is available under URL:

`https://solution.cloud/api/analysis/v2/wnnw97/_results`

Here wnnw97 is an example of the value of hash key.

Listing 31: Analysis Platform main API to gather files detected related to particular problem report

```
{"creation_time": "2022-06-06 07:12:23.833710", "log_bundles":
↪   [{"hash": "x993xo", "name": "ID652_details.json",
↪   "creation_time": "2022-06-06 07:12:25.227740", "lb_author":
↪   "bot", "as_author": "bot", "as_id": "95443"}, {"hash": "wnnw97",
↪   "name": "a1log_20220606_124937_b7a802b7.zip", "creation_time":
↪   "2022-06-06 07:12:24.680969", "lb_author": "bot", "as_author":
↪   "bot", "as_id": "95443"}, {"hash": "vyy89y", "name":
↪   "Snapshot_20220606-125158_724d11a3.zip", "creation_time":
↪   "2022-06-06 07:12:23.837496", "lb_author": "bot", "as_author":
↪   "bot", "as_id": "95443"}], "case_ids": []}
```

Listing 32: Analysis Platform detailed response for single log file

```
{"a6Analysis": {"status": "success", "config": {"imageName":
↪  "analysis/a6-analysis:1.0.9", "alias": "A6Analysis", "frontend":
↪  {"title": "A6 Analysis", "description": "A6 Analysis shows
↪  selected information related to A6 logs. For example carriers
↪  configurations, CPRI link states, chosen data from EEPROM, LED
↪  states.", "version": "1.0.9", "bundleUrl": "https://artifactory/⌋
↪  analysis-local/A6Analysis/1.0.9/A6Analysis.js", "tags":
↪  ["#basic"], "topLevelObjectName": "A6Analysis", "authorEmail":
↪  "", "mailingGroup": "", "organizationName": "", "teamName": "",
↪  "repo": "https://analysis/a6-analysis/tags/1.0.9"},
↪  "requirements": {"log": "none", "snapshot": "none",
↪  "matchingFilePaths": ["\\.zip$"]}}, "results": {"version": 1,
↪  "meaningful": true, "executive_summary": {"status": "ok",
↪  "summary_info": [{"id": "RMOD_L_1", "hw_unit": "radio1",
↪  "sw_version": "", "serial_number": "", "pcode": ""}]}, "extra":
↪  {"bot": {"suspects": {"A6_B1": {"score": 0, "reason": "Based on
↪  ML prediction.", "level": "gic"}, "A6_B2": {"score": 0,
↪  "reason": "Based on ML prediction.", "level": "gic"}, "A6_B3":
↪  {"score": 0, "reason": "Based on ML prediction.", "level":
↪  "gic"}, "A6_B4": {"score": 1, "reason": "Based on ML
↪  prediction.", "level": "gic"}, "A6_B5": {"score": 0, "reason":
↪  "Based on ML prediction.", "level": "gic"}, "A6_B6": {"score":
↪  0, "reason": "Based on ML prediction.", "level": "gic"},
↪  "A6_B7": {"score": 0, "reason": "Based on ML prediction.",
↪  "level": "gic"}, "A6_B8": {"score": 0, "reason": "Based on ML
↪  prediction.", "level": "gic"}, "A6_B8": {"score": 0, "reason":
↪  "Based on ML prediction.", "level": "gic"}, "A6_B9": {"score":
↪  0, "reason": "Based on ML prediction.", "level": "gic"},
↪  "A6_RD_UOAM_WR1": {"score": 0, "reason": "Based on ML
↪  prediction.", "level": "gic"}}, "next_to_check": {}, "schema":
↪  {"name": "bot", "version": "1"}}}, "files": ["details.json"]},
↪  "promoted": false}}
```

B.3.2 Via standardized Bug Tracking Support System reports

B.3.2.1 Description

Bug Tracking Support System (BTSS) is a solution which is responsible for communication with NOKIA systems related to management and reporting of software bugs. It follows the current state of cases and makes actions like for instance transferring cases between responsible groups or printing suggestions. It collects data of external services which deliver suggestions. Users are able to check historical data of suggestions like in the example below B.3.2.2.

B.3.2.2 Example of usage

Additionally, user can get summarize reports for each case by getting reports with following command:

Listing 33: Command to gather summarized results

```
wget http://example1.cloud/reports/basic/investigators.csv.gz
```

An example of record of the following file is shown below:

Listing 34: Single record from csv which contains summarize report of predictions

```
;A6;process:investigator_result;- possible affected gic A6_B4 at
↪  score 1 with reason Based on ML
↪  prediction.;;Investigator_Analysis_Platform_A6;;true;;business;b⌋
↪  ot.monitoring.business_events_emitter;;"{"bot_extra":
↪  [{"schema": {"name": "bot", "version": "1"}, "suspects":
↪  {"A6_B1": {"score": 0, "reason": "Based on ML prediction.",
↪  "level": "gic"}, "A6_B2": {"score": 0, "reason": "Based on ML
↪  prediction.", "level": "gic"}, "A6_B3": {"score": 0, "reason":
↪  "Based on ML prediction.", "level": "gic"}, "A6_B4": {"score":
↪  1, "reason": "Based on ML prediction.", "level": "gic"},
↪  "A6_B5": {"score": 0, "reason": "Based on ML prediction.",
↪  "level": "gic"}, "A6_B6": {"score": 0, "reason": "Based on ML
↪  prediction.", "level": "gic"}, "A6_B7": {"score": 0, "reason":
↪  "Based on ML prediction.", "level": "gic"}, "A6_B8": {"score":
↪  0, "reason": "Based on ML prediction.", "level": "gic"},
↪  "A6_B9": {"score": 0, "reason": "Based on ML prediction.",
↪  "level": "gic"}, "A6_B10": {"score": 0, "reason": "Based on ML
↪  prediction.", "level": "gic"}, "A6_B11": {"score": 0, "reason":
↪  "Based on ML prediction.", "level": "gic"}}, "next_to_check":
↪  {}}, {"schema": {"name": "bot", "version": "1"}, "suspects": {},
↪  "next_to_check": {"A6_B7": {"score": 1, "reason": "Most probably
↪  a1log_20220606_124937.zip does not contain FRM logs.", "level":
↪  "gic"}}}]";"[{"score": 1, "subsystem": "AMBIGUOUS_SUBSYSTEM",
↪  "development_unit": "A6", "group_in_charge":
↪  "A6_B4"}]";"[{"development_unit": "A6", "group_in_charge":
↪  "A6_B7", "score": 1, "subsystem":
↪  "AMBIGUOUS_SUBSYSTEM"}]";"[{"subsystem": "AMBIGUOUS_SUBSYSTEM",
↪  "development_unit": "A6", "group_in_charge":
↪  "A6_B7"}]";A6_DEFAULT;ID652;"[{"subsystem":
↪  "AMBIGUOUS_SUBSYSTEM", "development_unit": "A6",
↪  "group_in_charge": "A6_B4"}]";bf6f176a-135e-4c48-a3c0-cbba59ae0c⌋
↪  af;bot;72911c979882d0cc;2022-06-06T07:23:40.227218Z;045cfb59d8b0⌋
↪  e4b64e65a6457730cf30;0deb9bc957cf38c1;1.118.1
```

B.3.3 Via not standardized solution for additional prototype results

B.3.3.1 Description

To avoid excessive resource usage, temporary was created additional way for providing results of prototypes for suggestions. They are build into log analyzer. Example of accessing is shown below B.3.3.2.

B.3.3.2 Example of usage

`https://solution.cloud/api/analysis/v2/wnnw97/a6Analysis/_logs`

Listing 35: Chosen part of log of log analyzer

```
2022-06-06T07:19:15.530921120Z DEBUG -
↪  https://internal-predictions-analyzer-prototype-2 "POST
↪  /gic/predict-proba-with-classes HTTP/1.1" 200 413
2022-06-06T07:19:15.538384116Z INFO - External service
↪  https://internal-predictions-analyzer-prototype-3 response
↪  results: {'A6_B1': 0, 'A6_B2': 0, 'A6_B3': 0, 'A6_B4':
↪  0.1666666667, 'A6_B5': 0.33333333330000003, 'A6_B6': 0, 'A6_B7':
↪  0, 'A6_B8': 0.33333333330000003, 'A6_B9': 0, 'A6_B10': 0,
↪  'A6_B11': 0.1666666667}
2022-06-06T07:19:15.548701454Z INFO - External service
↪  https://internal-predictions-analyzer-prototype-1 response
↪  results: {'A6_B1': 0.0081865325, 'A6_B2': 0.0044039579, 'A6_B3':
↪  0.3379770554, 'A6_B4': 0.19636470220000002, 'A6_B5':
↪  0.1419945044, 'A6_B6': 0.012377810900000001, 'A6_B7':
↪  0.0025201137000000003, 'A6_B8': 0.22617901440000002, 'A6_B9':
↪  0.039268514500000004, 'A6_B10': 0.0065098135, 'A6_B11':
↪  0.0242179805}
2022-06-06T07:19:15.550666707Z INFO - External service
↪  https://internal-predictions-analyzer-prototype-2 response
↪  results: {'A6_B1': 0.0454744734, 'A6_B2': 0.0456472114, 'A6_B3':
↪  0.22147178650000002, 'A6_B4': 0.0853524283, 'A6_B5':
↪  0.22741401200000003, 'A6_B6': 0.0456536524, 'A6_B7':
↪  0.045453392, 'A6_B8': 0.1268815547, 'A6_B9': 0.0645140409,
↪  'A6_B10': 0.0453951545, 'A6_B11': 0.0467423052}
2022-06-06T07:19:15.734188522Z INFO - analysis - end
```

REFERENCES

[1] SMARTBEAR. Api endpoints - what are they? why do they matter?, June 2022. "https://smartbear.com/learn/pea1ormance-monitoring/api-endpoints/".

Appendix C

Implementation Attestation

NOKIA

ATTESTATION

The purpose of this document is to confirm that Łukasz Chmielowski has made improvements in the company's systems related to assignment of software bug reports by introducing various innovative solutions, including those described in the published articles:

- L. Chmielowski and M. Kucharzak. Impact of software bug report preprocessing and vectorization on bug assignment accuracy. In M. Choraś, R. S. Choraś, M. Kurzyński, P. Trajdos, J. Pejaś, and T. Hyla, editors, Progress in Image Processing, Pattern Recognition and Communication Systems, pages 153–162, Cham, 2022. Springer International Publishing.
- L. Chmielowski, P. Konstantynov, R. Luczak, M. Kucharzak, and R. Burduk. A novel method for software bug report assignment. Reliability Theory and Applications.
- L. Chmielowski, M. Kucharzak, and R. Burduk. Novel method of building train and test sets for evaluation of machine learning models related to software bugs assignment. Scientific Reports, 13, 12 2023.

Currently, work is underway to introduce:

- L. Chmielowski, M. Kucharzak, and R. Burduk. Application of explainable artificial intelligence in software bug classification. Informatyka, Automatyka, Pomiary w Gospodarce i Ochronie Srodowiska, 13(1):14–17, Mar. 2023.
- L. Chmielowski, P. Konstantynov, R. Luczak, M. Kucharzak, and R. Burduk. Application of multimodal neural networks in solving problem of labeling bug reports. Unpublished material.

Attested by:

[illegible]

Radosław Gawlik
R&D Director

Attested by:

[illegible]

Jakub Kalinowski

Person responsible for PhD coordination

Attested by:

[illegible]

Michał Kucharzak

PhD supervisor from company