Challenges in using Machine Learning to Support Software Engineering

Olimar Teixeira Borges[a], Julia Colleoni Couto[b], Duncan Ruiz[c]
and Rafael Prikladnicki[d]
PUCRS University, Porto Alegre, RS, Brazil

Keywords: Software Engineering, Machine Learning, Systematic Literature Review.

Abstract: In the past few years, software engineering has increasingly automating several tasks, and machine learning tools and techniques are among the main used strategies to assist in this process. However, there are still challenges to be overcome so that software engineering projects can increasingly benefit from machine learning. In this paper, we seek to understand the main challenges faced by people who use machine learning to assist in their software engineering tasks. To identify these challenges, we conducted a Systematic Review in eight online search engines to identify papers that present the challenges they faced when using machine learning techniques and tools to execute software engineering tasks. Therefore, this research focuses on the classification and discussion of eight groups of challenges: data labeling, data inconsistency, data costs, data complexity, lack of data, non-transferable results, parameterization of the models, and quality of the models. Our results can be used by people who intend to start using machine learning in their software engineering projects to be aware of the main issues they can face.

1 INTRODUCTION

Software Engineering (SE) projects are becoming increasingly complex, mainly due to the quantity and diversity of elements and their interactions (Boscarioli et al., 2017a). Designing, building, and testing complex software is challenging and requires people to work together, besides several tools and different techniques needed throughout the process (Butler et al., 2019). Artificial Intelligence (AI) offers various technologies to help deal with this complexity, including reasoning, problem-solving, planning, and learning, among others. Machine Learning (ML), more specifically, is a paradigm based on algorithms that learn from a previous experience (Mitchell, 1997): learning happens from knowing the data.

The software industry's development has made it possible to create increasingly extensive and more complete databases containing project data. To analyze this massive amount of data, the use of ML methods and techniques is beneficial (Boscarioli et al., 2017b; Borges et al., 2020). For this reason, SE has shown increasing interest in intensifying the use of AI and ML (Beal. et al., 2017; Amershi et al., 2019) to take advantage of this data.

This paper aims to map, describe, and better understand the most frequent issues faced by teams that decide to implement ML techniques and tools to support their SE projects. For that, we developed a Systematic Literature Review (SLR), based on (Kitchenham and Charters, 2007) process, and we used the PICO (Murdoch, 2018) to assist in constructing the research question. We conducted the literature search in 2019.

As a result of the SLR, we found 27 papers that report challenges encountered while using ML in the SE domain. We grouped the challenges by similarity and then categorized them into eight groups of challenges: data labeling, data inconsistency, data costs, data complexity, lack of data, non-transferable results, parameterization of the models, and quality of the models.

This paper presents the following structure: Section 2 addresses the methodology we used in this work; Section 3 displays the paper's results, the classification we carried out, and its discussions; and, finally, Section 4 concludes and provides suggestions for future work.

[a] https://orcid.org/0000-0002-2567-2570
[b] https://orcid.org/0000-0002-4022-0142
[c] https://orcid.org/0000-0002-4071-3246
[d] https://orcid.org/0000-0003-3351-4916

Borges, O., Couto, J., Ruiz, D. and Prikladnicki, R.
Challenges in using Machine Learning to Support Software Engineering.
DOI: 10.5220/0010429402240231
In *Proceedings of the 23rd International Conference on Enterprise Information Systems (ICEIS 2021) - Volume 2*, pages 224-231
ISBN: 978-989-758-509-8

2 METHODOLOGY

We performed a SLR to map the literature on the challenges in the use of ML in SE tasks. Literature reviews or mappings are often used to structure a research area, map and classify topics to find patterns and research gaps. According to (Kitchenham and Charters, 2007), a SLR has to present the processes to guide planning, conducting, and reporting the review. We present each phase and its sub-processes henceforward.

To start, we develop a research question (RQ) to guide the process and help us stay aligned with the scope: **RQ1: What are the challenges when using machine learning in software engineering projects?** Based on the RQ, we used the PICO method (Murdoch, 2018) to elaborate the search string: **P**opulation: Software Engineering. **I**ntervention: Machine Learning. **C**omparison: - **O**utcome: Challenges in using ML in SE.

Therefore, the search string we applied to online search engines is composed of the terms *("software engineering" AND "machine learning" AND (challenge OR threat))*. We searched for these terms in the title, abstract, or keywords of the papers.

During the SLR protocol development, we performed a random search on the web to identify a paper that would be known to answer the research question to use as a control paper. We use control paper to test the search string, as it must be among the online search engines' results. We choose this paper based on reading and analyzing some papers to verify adherence to the theme — our control paper is: (Hamouda, 2015).

2.1 Study Screening

We defined inclusion and exclusion criteria to assist in the papers' selection process. Each paper is assigned at least one criterion, and the papers that we accepted meet all of the following inclusion criteria: (I1) Qualitative or quantitative research about software engineering projects that use machine learning tools and techniques and present related challenges; (I2) Complete study available in electronic format. (I3) Paper published in conference proceedings or journals.

We identified 301 papers at the beginning of the review process. For each paper, we performed the first filter (Selection phase) to identify their RQ adherence, in which we read the title, the abstract, and the keywords. During the data extraction phase, we accepted 37 papers, which passed through a second full reading filter. At the end of the process, we rejected 274 papers due to the exclusion criteria presented in Table 1.

Table 1: Number of papers rejected due to exclusion criteria.

Exclusion Criteria	Amount
(E1) Incomplete or short paper (up to 3 pages)	8
(E2) Paper unavailable for download	1
(E3) Paper does not answer research questions	158
(E4) Duplicate study	51
(E5) Conference Proceedings Index	38
(E6) Book or book chapter	10
(E7) Literature review or mapping	8
Number of Papers Rejected:	**274**

We used the StArt (State of the Art through systematic review) tool (Fabbri et al., 2016) to assist in the extraction and compilation of data. After finishing the data extraction, we exported the results to Google Sheets, to facilitate collaborative online work among researchers. Finally, we accepted 27 papers in the final set, which answer the research question and meet the inclusion criteria.

3 RESULTS

In this section, we present the classification of the 27 papers that we accepted and the challenges we extracted from these papers. We applied the search string in August 2019 in the following search engines: ACM, IEEE, Springer, arXiv, Science Direct, Web of Science, Google Scholar, and Scopus. The Table 2 shows the number of papers by search engines. Scopus is the engine where we retrieved most of the results, being the origin of 1/3 of the accepted papers, and this is because it indexes several other bases.

Table 2: Papers by search engines.

Source	Initial	Selection	Accepted
Scopus	137	12	9
IEEE Xplore	13	0	0
Web of Science	21	2	2
ACM	31	8	7
Springer	69	9	4
arXiv	8	3	2
Science Direct	5	1	1
Google Scholar	17	2	2
Accepted Papers:			**27**

We analyzed the country of origin of the papers that most often inform the challenges in their papers. This analysis shows us that the USA (14 researchers), Canada (12), Germany (10), China (7), India (5), and Brazil (5) are the origin of the most accepted papers. Note that several papers have more than one author, and we took it into account, so the amount added will be greater than the number of papers.

3.1 Data Extraction

During the information extraction phase, we read the 27 papers to analyze and classify them. Twelve papers are published in journals, and fifteen are published in computer science conferences and workshops. Most papers are from 2019 (8 papers) and 2018 (5 papers), with 89% published since 2012, suggesting that it is a rising topic. The oldest paper is from 2004 (Bowring et al., 2004). The final consultation period is August 31, 2019.

3.2 Classification Scheme

We created a classification scheme to present the results according to the mapped challenges. In this section, we present the papers' analysis and the answer to our RQ (*What are the challenges when using machine learning in software engineering projects?*).

We categorize the papers according to the similarity between the challenges they present. Five challenges are related to the data: data labeling, data inconsistency, data costs, data complexity, and lack of data. Three other challenges are related to ML models: non-transferable results, parameterization of the models, and quality of the models. Table 3 shows the classification of the challenges and the corresponding papers. Some papers present more than one challenge, so the sum of the papers presented in this table will be higher than the 27 papers accepted. Next, we present and discuss each challenges.

1. Data Labeling.

Description: Labeled data are data that have classified and tagged information. This information about the data helps to understand its patterns and behavior, which will be used for future analysis.

Example in SE: During the operation of a system, several records and reports on its operation are generated. Most of these records are for reporting errors and failures in execution. These reports usually present the type of error, the class's identification that generated the error, the code snippet, the date of the occurrence, and other related information. These data, labeled as failures, can continuously improve the system and predict possible error cases, improving the software development process.

Analysis of Papers: If we want to have more high-quality and assertive ML models, it is necessary to access a large amount of data for its training (Moataz A. Ahmed and Hamdi A. Al-Jamimi, 2013). However, in the SE, there is still a considerable shortage of datasets to be used in research in the area (Moataz A. Ahmed and Hamdi A. Al-Jamimi, 2013; Petkovic et al., 2014). For instance, the image recognition field, a sector in constant growth, is necessary to access a large amount of data. The lack of this labeled data ends up leading to the development of simulated scenarios to meet its needs (Bowring et al., 2004; Porru et al., 2016; Runeson, 2019).

In line with the scarcity of data in general, it is still necessary to deal with the lack of data labeled/tagged (Qiu et al., 2019). Also, access to labeled training data is still a limitation for many applications (Runeson, 2019). People are needed to carry out manual labeling, classification, and analysis, which is quite problematic and challenging (Amal et al., 2014; Bangash et al., 2019). One of the factors of this challenge is the complexity inherent to natural language, limiting the applicability of automated approaches (Kurtanović and Maalej, 2018).

2. Data Inconsistency.

Description: Inconsistent data can be considered as missing, incorrect, incomplete, duplicated, and unbalanced data. These data have not been prepared yet and need to be treated to become consistent and useful.

Example in SE: Historical system data can assist at the beginning of planning a software project. Historical data can guide the creation of software requirements. They can guide how to design system requirements but will likely need to undergo analysis and pre-process to be useful.

Analysis of Papers: ML models are considered reliable when they are trained from consistent data. However, data collection and validation for SE requires efficient processing techniques to handle large volumes of data, manage different inconsistencies and extract adequate resources from programs, to improve learning performance (Belsis et al., 2014; Phan et al., 2018). SE data rarely has a normal distribution (Bettenburg et al., 2015). For example, in the process of software requirements, due to their unique characteristics, the relevant data sets are highly dimensional, sparse and often result from ambiguous expressions and, therefore, end up presenting difficult challenges for data processing techniques (Belsis et al., 2014).

The vast majority of research in the area uses data collected from available software repositories. How-

Table 3: Papers *versus* challenge.

	Challenge	Papers
1	**Data Labeling:** 9 papers	(Moataz A. Ahmed and Hamdi A. Al-Jamimi, 2013), (Amal et al., 2014), (Bangash et al., 2019),(Bowring et al., 2004), (Kurtanović and Maalej, 2018), (Petkovic et al., 2014), (Porru et al., 2016), (Qiu et al., 2019), (Runeson, 2019)
2	**Data Inconsistency:** 9 papers	(Allamanis, 2018), (Barros et al., 2008), (Belsis et al., 2014), (Bettenburg et al., 2015), (Phan et al., 2018), (Petkovic et al., 2014), (Sharma et al., 2012), (Twala and Cartwright, 2010), (Ying and Robillard, 2013)
3	**Data Costs:** 5 papers	(Belsis et al., 2014), (Cui et al., 2019), (Phan et al., 2018), (Runeson, 2019), (Sharma et al., 2012)
4	**Data Complexity:** 3 papers	(Douthwaite and Kelly, 2017), (Hamouda, 2015), (Meyer and Gruhn, 2019)
5	**Lack of Data:** 3 papers	(Hesenius et al., 2019), (Porru et al., 2016), (Zhang et al., 2018)
6	**Non-transferable Results:** 5 papers	(Amal et al., 2014), (Hosni et al., 2018), (Kaur et al., 2019), (Porru et al., 2016), (Turhan, 2012)
7	**Parameterization of Models:** 3 papers	(Belsis et al., 2014), (Porru et al., 2016), (Zhang et al., 2018)
8	**Quality of the Models:** 2 papers	(Foidl and Felderer, 2019), (Meyer and Gruhn, 2019)

ever, there are many repositories with duplicate code, which constitutes data inconsistency (Allamanis, 2018). Still, code fragments are analyzed, which can present technical challenges because code fragments are not complete programs (Ying and Robillard, 2013). One of the big problems when working with inconsistent data is that after the data preparation phase for use in training machine learning models, the data set ends up becoming smaller (Barros et al., 2008) due to the need to deleting invalid data. There is also the issue of using unbalanced, missing, or incorrect training data (Petkovic et al., 2014; Sharma et al., 2012; Twala and Cartwright, 2010).

3. Data Costs.
Description: A large amount of data needs to be stored in suitable locations, requiring computational power for its processing, and having maintenance costs. All of these characteristics are related to the cost of the data. Whether financial or time, these costs need to be calculated according to each data set's needs.
Example in SE: Software systems can generate larges amounts of data about their operation and execution. These data need strategies to be stored and accessed when necessary. Likewise, it is essential to have computational processing to obtain important information about the system data.
Analysis of Papers: Data collection and validation for SE requires efficient processing techniques to deal with large volumes of data, which makes it possible to extract adequate, relevant, and non-redundant resources from programs to improve learning performance (Belsis et al., 2014; Cui et al., 2019; Phan et al., 2018). With an extensive data set, processing can become costly, and therefore it is essential to perform an appropriate resource selection (Sharma et al., 2012). However, data is not always available in an open and public way, and there is no exact model for sharing or monetizing data for use in ML models. As a result, there is an increase in the demands and costs of collecting, storing, and sharing SE data for ML (Runeson, 2019) applications.

4. Data Complexity.
Description: The data's complexity can be related to its diversity, the type of data, the size, among other factors.
Example in SE: Data from software development projects is usually textual, considering source code, description of requirements, data resulting from automated tests. These data, often not standardized,

present an inherent complexity that impacts ML models' creation and validation.

Analysis of Papers: During the requirements phase, use-case modeling has a characteristic complexity. ML models emerge to assist in this phase with the development of knowledge-based systems and ontologies, aiming at the automation of requirements management and modeling of problem domains (Hamouda, 2015). However, one of the biggest challenges in training these learning methods for real-world problems is so precisely how to effectively manage the complexity of the use cases (Meyer and Gruhn, 2019). ML models are challenging to use at the beginning of a project's life cycle when it aims to develop a set of specific data and knowledge requirements about the project, only from the use cases (Douthwaite and Kelly, 2017).

5. Lack of Data.

Description: In the analysis and use of ML techniques on any data, it is essential first to understand these data, know its meaning, the type of data, the size, and other characteristics, even to know if the results of the model make sense.

Example in SE: In this context, it is essential to use domain experts, such as engineers, analysts, or software architects, to understand the data.

Analysis of Papers: In the age of Big Data, large data sets are created every moment around the world. However, to extract useful information from these large data sets, it is necessary to learn how to process them. Therefore, knowledge about these data's domain is essential, and in SE, this is no different. For developers and data analysts to create efficient and useful ML solutions, specific knowledge about the data is required, either in the data analysis phase or selecting the most appropriate algorithms for each type of solution (Hesenius et al., 2019).

ML models must be evaluated and validated to be considered efficient and useful. Such models are often used in many real-world applications, assessing systems' suitability and validating their implementations against user requirements and specific project scenarios. However, the lack of *a priori* knowledge of the data ends up becoming a significant challenge in the area (Porru et al., 2016; Zhang et al., 2018).

6. Non-transferable Results.

Description: It deals with the construction of models that work very well to solve a specific problem in one domain. However, it cannot be transferred to deal with the same problem in another domain, mainly due to the data's dissimilarities.

Example in SE: Software development projects have several peculiarities. For example, a company that develops software for other client companies. Suppose an ML model was developed based on the user stories of a project for a specific client, and there is no standardization between fields between one client and another. In that case, the same ML model cannot be applied directly to both clients without changes in the model.

Analysis of Papers: There is still an absence of related work that uses machine learning techniques focused on refactoring software (Amal et al., 2014). Machine learning models that make use of regression and classification need historical data to be trained. However, each project tends to be unique in its specifications, and ML models tend to generalize their results and conclusions (Hosni et al., 2018; Kaur et al., 2019). This generalization can be a problem because when we use generalized models in particular projects, it can generate assumptions of a causal relationship between the data (Porru et al., 2016). As a result, the challenge of generating models with non-transferable results in different projects and studies remains (Turhan, 2012).

7. Parameterization of Models.

Description: The ML algorithms need to be configured with some parameters before their initialization. In a clustering algorithm, the parameters include the number of clusters generated based on the data and a distance function (Euclidean, Manhattan, and others).

Example in SE: Several more suitable clusters may be necessary to group tickets related to software incidents, classifying them in new functionality, bug, or duplicate.

Analysis of Papers: One of the fundamental steps when creating ML models using SE data is the correct choice of metrics and parameters. During the training of a classification model, for example, if specific attributes are used in an arbitrary manner, such as algorithms and other parameters, an incorrect model will probably be generated, presenting classification errors (Porru et al., 2016). Such a challenge requires experience in creating models, knowledge of the data, and more efficient techniques/algorithms for each task. This is the cluster classification case, which tries to discover hidden patterns in the data without prior knowledge about them. Any wrong setting in the parameters will make it difficult to evaluate the quality of the results and the ideal number of clusters (Belsis et al., 2014; Zhang et al., 2018).

8. Quality of the Models.

Description: It refers to the accuracy of the model results to how well these models get it right.

Example in SE: Using the same example of classification of incidents related to the software under development, it would be the percentage of incidents classified as new functionalities and that deal with requirements for implementing new functionalities in the software.
Analysis of Papers: There is a constant search for guarantee and quality evaluation of ML models. Maintaining the quality of these learning models, applied to SE, is still a very challenging task (Meyer and Gruhn, 2019). The development of these models at a mature level and that maintains quality and productivity goals is not trivial (Foidl and Felderer, 2019). Such models have a high degree of complexity, becomes challenging to validate the data used in their training to assess their quality (Meyer and Gruhn, 2019).

3.3 Discussion

Eight of the papers contemplate more than one challenge at the same time (Amal et al., 2014; Belsis et al., 2014; Meyer and Gruhn, 2019; Petkovic et al., 2014; Porru et al., 2016; Runeson, 2019; Sharma et al., 2012; Zhang et al., 2018). Among them, the most reported challenges are data labeling, data inconsistency, and parameterization of models. As labeling was one of the most reported challenges, it can be an exciting point for exploration. For machine learning models to become efficient, the data must be correct. Also, adequately labeled data makes initial data preprocessing work faster.

The data's inconsistency was also one of the most mentioned challenges, and it covers the part of missing, inconsistent, and duplicate data. This inconsistency causes researchers, developers, and data analysts to waste time processing the data before using it as input to their ML models. The challenge related to the parameterization of models deals directly with the configuration of ML models. The correct parameters and algorithms for elaborating the model require knowledge and experience about ML data and techniques. Each domain is specific and needs to be analyzed in depth so that the appropriate parameters are selected to generate an appropriate model.

In Table 4, we list the eight challenges, alongside some examples of tasks that are affected by the challenge and examples of the ML techniques and algorithms addressed in the papers we accepted. For each challenge, the papers present the environment configurations where the challenges were identified or present configurations explored through experiments. These configurations are formed by ML technique(s) or algorithm(s) applied to some SE task(s).

Many of the reported ML algorithms are part of traditional ML; however, five of the challenges discussed at least one artificial neural network (ANN) technique, which reveals that this technique is being increasingly used to solve SE problems. However, at the same time, ANN is more linked to the generation of challenges when applied to some of the SE's tasks, mainly about the data-related challenges. We also noticed that the tasks presented are mostly related to software source codes, in the analysis, searching for defects or code smells, and refactoring. That is, there is an implicit need to solve tasks related to the source code using ML.

The discussion of these eight challenges aims to benefit software development teams who want to start using ML techniques and tools. The intention is that they can become aware of the main challenges and create prevention strategies before starting their projects. Also, SE teams that already use ML in their processes can use the challenges to identify possible problems in the planning and executing of their systems. In this way, we seek to minimize recurring problems and reduce the time spent with the impediments resulting from these challenges.

4 CONCLUSION

In this paper, we presented a systematic literature review developed to map the main challenges reported in scientific papers that deal with ML tools and techniques as support to software engineering. To do so, we queried eight of the leading online search engines for Computer Science papers. At the end of the SLR, we accepted 27 papers.

We identified eight main challenges during the papers' analysis, five related to the data itself, and three related to ML models. The identified challenges demonstrate the importance of consistency, labeling, and understanding of the data used as input to ML techniques so that the results can be useful. The challenge related to the costs of acquiring, storing, and processing this data is also highlighted. Also, parameterization and model quality are topics of attention when using ML. Another point of attention is the difficulties in replicating some models, which is justified by each software project's specifics.

Like all qualitative research, this paper has some limitations. In the selection phase, we did not define an initial year for the search for the papers. This could lead to identifying old challenges, which currently no longer qualify as challenges, as they could have already been resolved. As it is essential to understand the challenges from the beginning, this is an accept-

Table 4: List of SE Tasks and ML Techniques/Algorithms per challenge.

Challenge	SE Tasks	Techniques/ML Algorithms
1. **Data Labelling**	Code Analysis; Software Defects; Software maintenance;	Artificial Neural Networks; Bayesian Networks; Decision Tree; Gaussian Process Classifier; Latent Dirichlet Allocation; Logistic Regression; Multivariate Adaptive Regression Splines; Naïve Bayes; Random Forest; Support Vector Machine;
2. **Data Inconsistency**	Code Analysis; Data analysis; Estimated Software Effort; Estimation of Software Metrics; Code Summary;	Artificial Neural Networks; Multivariate Adaptive Regression Splines; Naïve Bayes; Random Forest; Support Vector Machine;
3. **Data Costs**	Program Classification; Code Smells in Design Templates; Software Failure; Requirements Validation;	Bayesian Networks; Decision Tree; Naïve Bayes; Random Forest; Support Vector Machine;
4. **Data Complexity**	Requirements Analysis; System Security Analysis;	Artificial Neural Networks; Bayesian Network; Classification Trees; Genetic Algorithms; Logistic Regression;
5. **Lack of Data**	Detection of Code Anomalies; Estimated Software Effort;	Artificial Neural Network; Naïve Bayes; Support Vector Machine;
6. **Non-transferable Results**	Code Smell detection; Code Refactoring;	k-means; Naïve Bayes; Support Vector Machine;
7. **Parameterization of Models**	Systems Validation and Evaluation;	Artificial Neural Network; Naïve Bayes; Support Vector Machine;
8. **Quality of the Models**	Component-based Software Development; Software Quality;	Hierarchical Reinforcement Learning;

able limitation. To minimize the risk of not having a broad and diverse number of documents, we searched on eight well-known search engines on the web. To reduce the researchers' bias, two researchers worked in parallel, and two other researchers helped in case of disagreement.

For future work, we want to map possible solutions for each challenge. We also aim to search for reports and experiments that address and explain how each challenge is addressed and within which SE area they most persist. This search will help the community to find possible solutions and strategies to minimize their problems and solve them within each phase of a project's development. Furthermore, this study did not aim to validate whether the ML techniques, algorithms, and tools presented in the discussion of the results, are useful for using linked SE tasks, as shown in Table 4. We reported and structured this information in this study. However, it is up to future work to carry out experiments that could validate our presented approaches and configurations.

ACKNOWLEDGEMENTS

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nivel Superior – Brasil (CAPES) – Finance Code 001, FAPERGS (17/2551-0001/205-4), and CNPq.

REFERENCES

Allamanis, M. (2018). The adverse effects of code duplication in machine learning models of code. *CoRR*, abs/1812.06469.

Amal, B. et al. (2014). On the use of machine learning and search-based software engineering for ill-defined fitness function: A case study on software refactoring. In *SSBSE*, pages 31–45. Springer.

Amershi, S. et al. (2019). Software engineering for machine learning: a case study. In *ICSE*, pages 291–300. IEEE.

Bangash, A. A. et al. (2019). What do developers know about machine learning: A study of ml discussions on stackoverflow. In *MSR*, pages 260–264. IEEE Press.

Barros, R. C. et al. (2008). Issues on estimating software metrics in a large software operation. In *SEW*, pages 152–160.

Beal., F., de Bassi., P. R., and Paraiso., E. C. (2017). Developer modelling using software quality metrics and machine learning. In *ICEIS*, pages 424–432.

Belsis, P., Koutoumanos, A., and Sgouropoulou, C. (2014). Pburc: a patterns-based, unsupervised requirements clustering framework for distributed agile software development. *RE*, 19(2):213–225.

Bettenburg, N., Nagappan, M., and Hassan, A. E. (2015). Towards improving statistical modeling of software engineering data: think locally, act globally! *ESE*, 20(2):294–335.

Borges, O. T., Couto, J. C., Ruiz, D., and Prikladnicki, R. (2020). How machine learning has been applied in software engineering? In *ICEIS*, pages 306–313.

Boscarioli, C., Araújo, R., and Maciel, R. (2017a). I grandsi-br–grand research challenges in information systems in brazil 2016-2026. *CE-SI - SBC*.

Boscarioli, C., Araújo, R., and Maciel, R. (2017b). I grandsi-br–grand research challenges in information systems in brazil 2016-2026. *CE-SI - SBC)*.

Bowring, J. F., Rehg, J. M., and Harrold, M. J. (2004). Active learning for automatic classification of software behavior. *SIGSOFT SEN*, 29(4):195–205.

Butler, C. W., Vijayasarathy, L. R., and Roberts, N. (2019). Managing software development projects for success: Aligning plan- and agility-based approaches to project complexity and project dynamism. *PM journal*, 0(0):8756972819848251.

Cui, C., Liu, B., and Li, G. (2019). A novel feature selection method for software fault prediction model. In *RAMS*, pages 1–6.

Douthwaite, M. and Kelly, T. (2017). Establishing verification and validation objectives for safety-critical bayesian networks. In *ISSREW*, pages 302–309.

Fabbri, S. et al. (2016). Improvements in the start tool to better support the systematic review process. In *EASE*, pages 1–5.

Foidl, H. and Felderer, M. (2019). Risk-based data validation in machine learning-based software systems. In *MaLTeSQuE*, pages 13–18. ACM.

Hamouda, A. (2015). New trends in learning for software engineering. In *IC-RACE*. Atlantis Press.

Hesenius, M. et al. (2019). Towards a software engineering process for developing data-driven applications. In *RAISE*, pages 35–41. IEEE Press.

Hosni, M. et al. (2018). On the value of parameter tuning in heterogeneous ensembles effort estimation. *Soft Computing*, 22(18):5977–6010.

Kaur, A., Jain, S., and Goel, S. (2019). Sp-j48: a novel optimization and machine-learning-based approach for solving complex problems: special application in software engineering for detecting code smells. *NCA*.

Kitchenham, B. and Charters, S. (2007). Guidelines for performing systematic literature reviews in software engineering. Technical Report EBSE-2007-01, Department of Computer Science, University of Durham.

Kurtanović, Z. and Maalej, W. (2018). On user rationale in software engineering. *RE*, 23(3):357–379.

Meyer, O. and Gruhn, V. (2019). Towards concept based software engineering for intelligent agents. In *RAISE*, pages 42–48. IEEE Press.

Mitchell, T. M. (1997). *Machine learning*. McGraw hill.

Moataz A. Ahmed and Hamdi A. Al-Jamimi (2013). Machine learning approaches for predicting software maintainability: a fuzzy-based transparent model. *IET Software*, 7(6):317–326.

Murdoch, U. (2018). Systematic reviews: Using pico or pico. https://goo.gl/fqPoCY. Accessed: 2018-12-20.

Petkovic, D. et al. (2014). Setap: Software engineering teamwork assessment and prediction using machine learning. In *FIE*, pages 1–8.

Phan, A. V. et al. (2018). Automatically classifying source code using tree-based approaches. *DKE*, 114:12–25.

Porru, S. et al. (2016). Estimating story points from issue reports. In *PROMISE*, pages 2:1–2:10. ACM.

Qiu, S. et al. (2019). Cross-project defect prediction via transferable deep learning-generated and handcrafted features. In *ICSE*, pages 431–436.

Runeson, P. (2019). Open collaborative data - using oss principles to share data in sw engineering. In *ICSE*, pages 25–28.

Sharma, M. et al. (2012). Predicting the priority of a reported bug using machine learning techniques and cross project validation. In *ISDA*, pages 539–545.

Turhan, B. (2012). On the dataset shift problem in software engineering prediction models. *ESE*, 17(1):62–74.

Twala, B. and Cartwright, M. (2010). Ensemble missing data techniques for software effort prediction. *Intell. Data Anal.*, 14(3):299–331.

Ying, A. T. and Robillard, M. P. (2013). Code fragment summarization. In *FSE*, pages 655–658. ACM.

Zhang, Z. et al. (2018). A validation and quality assessment method with metamorphic relations for unsupervised machine learning software. *CoRR*, abs/1807.10453.