How Machine Learning Has Been Applied in Software Engineering?

Olimar Teixeira Borges[a], Julia Colleoni Couto[b], Duncan Dubugras A. Ruiz[c]
and Rafael Prikladnicki[1] [d]
School of Technology, PUCRS, Porto Alegre, Brazil

Keywords: Software Engineering, Machine Learning, Mapping Study.

Abstract: Machine Learning (ML) environments are composed of a set of techniques and tools, which can help in solving problems in a diversity of areas, including Software Engineering (SE). However, due to a large number of possible configurations, it is a challenge to select the ML environment to be used for a specific SE domain issue. Helping software engineers choose the most suitable ML environment according to their needs would be very helpful. For instance, it is possible to automate software tests using ML models, where the model learns software behavior and predicts possible problems in the code. In this paper, we present a mapping study that categorizes the ML techniques and tools reported as useful to solve SE domain issues. We found that the most used algorithm is Naïve Bayes and that WEKA is the tool most SE researchers use to perform ML experiments related to SE. We also identified that most papers use ML to solve problems related to SE quality. We propose a categorization of the ML techniques and tools that are applied in SE problem solving, linking with the Software Engineering Body of Knowledge (SWEBOK) knowledge areas.

1 INTRODUCTION

Machine Learning (ML) is an Artificial Intelligence (AI) sub-field composed of several algorithms for approximating patterns in data discovery (Mitchell, 1997). ML-based systems learn from experience, and its algorithms have proved to be of great practical importance to automate several Software Engineering (SE) tasks, such as refactoring (Kumar et al., 2019), software defects detection (Al-Nusirat et al., 2019), and effort estimation (Choetkiertikul et al., 2018).

SE projects are highly complex and often unpredictable because they depend on interrelated factors to succeed (De Rezende et al., 2018). Mostly, software development projects depend on people and the tools they have to work. As we know, human actions are subject to failure, mainly because of the inherent differences related to knowledge and experience (Anu et al., 2018).

The unpredictability inherent to software projects can lead to time-wasting, mostly related to rework (Sedano et al., 2017), which increases the project cost. There are a lot of SE tasks that can be automated to reduce human effort and project cost. However, to automate these tasks using ML, we need to have the SE project's data to explore. Fortunately, there is a vast amount of software project data available in public repositories. This data can be widely used to extract knowledge and improve efficiency and quality in software development projects (Liu et al., 2018). We can use a variety of ML techniques for each type of SE problem. However, to effectively use these techniques, we need to have a clear understanding of what are the problems with SE, what are the problems 5 that can be solved using ML, and what are the techniques and tools that can be used (Zhang and Tsai, 2003).

In this paper, we report how ML techniques and tools are used in SE projects, according to the analysis of scientific papers. We also map the main SE problems that are being solved using ML. To do so, we perform a mapping study, searching for papers in eight different electronic databases. We follow the steps suggested by (Kitchenham and Charters, 2007), and we use PICO (Murdoch, 2018) to formulate the research question. Additionally, we apply the Cohen Kappa (Landis and Koch, 1977) method to measure and help to improve the agreement between the researchers. To reduce ambiguity and use a well-established guide (Duarte, 2019), we adopted the

[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.
How Machine Learning Has Been Applied in Software Engineering?.
DOI: 10.5220/0009417703060313
In *Proceedings of the 22nd International Conference on Enterprise Information Systems (ICEIS 2020) - Volume 2*, pages 306-313
ISBN: 978-989-758-423-7

Guide to the SWEBOK (Bourque et al., 2014) Knowledge Areas (KA) to categorize SE domain issues.

We found 177 papers that present solutions based on ML for domain issues in SE. We discovered the most commonly used techniques, composed by algorithms, libraries, and programming languages, and we also map the tools most researchers report to use during its experiments. We also mapped the 86 SE domain issues most frequently benefited from using ML. These domains include prediction of software design defects and failures, effort and cost estimation, and SE requirements classification.

2 MATERIALS AND METHODS

Systematic literature reviews or mappings are often used to structure a research area, mapping, and classifying topics to find research patterns and gaps (Petersen et al., 2015). According to (Kitchenham and Charters, 2007), a literature mapping is composed of three main processes, named planning, conducting, and reporting. Each process has subsequent steps, which we follow and present from now on.

2.1 Research Questions

We developed three Research Questions (RQ) to investigate the application of ML in SE domain issues:
RQ1: *How has machine learning been used in software engineering?*
RQ2: *Which software engineering domain issues can benefit most from using machine learning?*
RQ3: *What computational machine learning techniques and tools are most used to solve software engineering problem domains?*

Based on the research questions, we used the PICO technique to elaborate the first search string, searching for the terms *(”software engineering” AND ”machine learning” AND ”tool” AND ”technique” AND ”domain”)* in the papers title, abstract, or keywords. Using this string, we got 61 papers in the final accepted set. Because we aim to broadly map the literature about ML for SE, we decided to alter the search string so that we could improve the number of papers accepted. Then we searched only for the terms *(”software engineering” AND ”machine learning”)* in the same places. Thus, we accepted 177 papers in the second string, 116 more than in the first one. For this reason, we kept the results we obtained from the second search string.

To answer RQ1, we classify the papers using two approaches: one automatically clustering the papers using ML and another using the SWEBOK KA. In the first classification approach, we use ML techniques to cluster the papers we accepted using Orange tool [1]. We create a Corpus containing the title, abstract, and keywords and used it as input for Orange. For text preprocessing, we use the WordNet Lemmatizer parameter for normalization, remove regular expressions and stopwords, select only words that are 30-90% frequency and only the 500 most frequent tokens. Along with stopwords, we removed words that appear in all papers and their radicals, such as ”*soft*”, ”*softw*”, ”*softwar*”, ”*software*”, ”*thi*”, ”*use*”, ”*using*”, ”*learn*”, ”*learning*”, ”*machin*”, ”*machine*”, ”*engineer*”, and ”*engineering*”. Then, we created a Bag of Words and set the Document Frequency parameter to IDF (Inverse Document Frequency), which measures the importance of a word in a given document. We used the Silhouette coefficient from K-means to identify an optimal amount of clusters. The best score was 0.236 for 2 clusters. Then we use the Cosine's Distances to create the clusters. Lastly, we create a Hierarchical Clustering to represent the data visually.

Then, in our second approach, we classified the papers manually using the ten SWEBOK KA. We use SWEBOK used because it is a well recognized and widely accepted standard in the SE community. Table 1 presents the SWEBOK KA.

For answering RQ2, we consider domain issues as the contexts and tasks in SE the papers explore, such as software defect detection, software behavior, and reverse engineering. To answer RQ3, we use the definitions of the terms provided by International Standard - Systems and software engineering – Vocabulary (ISO/IEC/IEE, 2010) for the terms ”tools” and ”techniques”. According to the Standard, a tool is a software product used to help in some activity, while a technique is a systematic procedure or method, such as an algorithm. Therefore, in this paper, ML techniques are any algorithm or statistical methods used to make predictions, classifications, and clustering.

2.2 Study Screening

To increase the quality of our findings, we define inclusion and exclusion criteria. We associate each paper to at least one criterion. The papers we select match all the following criteria:

- **(I1)** Qualitative or quantitative research about SE using ML.
- **(I2)** Presents complete study in electronic format.
- **(I3)** Conference or journal paper.

[1] https://orange.biolab.si

Table 1: 10 SWEBOK (Bourque et al., 2014) Knowledge Areas (KA).

Knowledge Areas (KA)	**Description**
1. Software Requirements	Comprise requirements process, elicitation, analysis, specification, and validation.
2. Software Design	Present processes to create the software structure and architecture, introduces how to create, analyze, and evaluate user interface design.
3. Software Construction	Presents tools and technologies for software development.
4. Software Testing	Addresses test levels, tools, techniques, processes, and test-related measures.
5. Software Maintenance	Comprise processes, techniques, and main issues related to software development maintenance.
6. Software Configuration Management	Presents how to plan and control the software development environment.
7. Software Engineering Management	Introduces processes related to software project management, such as planning, scope definition, enactment, review and evaluation, closure, and how to measure the project progress.
8. Software Engineering Process	Addresses process definition, software life cycles, how to asses and improve software projects, and tools to be used in the SE process.
9. Software Engineering Models and Methods	Presents the types of models, how to analyze the models, and SE methods, such as heuristic, formal, prototyping, and agile.
10. Software Quality	Introduces processes to manage and maintain the software quality process.

Table 2: Number of papers rejected due to exclusion criteria.

Exclusion Criteria	Amount
(E1) Short paper ($\leqslant$ 3 pages)	28
(E2) Paper is unavailable for download	4
(E3) Paper non matching at least one RQ	451
(E4) Duplicate paper	0
(E5) Written in a language other than English	1
(E6) Conference proceedings index	100
(E7) Book or book chapter	24
(E8) Literature Review or Mapping	5

Table 3: Kappa results, based on (Landis and Koch, 1977).

Kappa values	Agreement	1st	2nd	3rd	4th
<0	Poor				
0 - 0.20	Slight				
0,21-0,40	Fair	0.38	0.27		
0,41-0,60	Moderate			0.55	
0,61-0,80	Substantial				
0,81-1	Almost perfect				0.82

We started the study having 1314 papers to analyze. For all papers, we read the title, abstract, and keywords. From the 1314 papers, 522 are duplicated, and we accepted 177 papers in the final set. Among the 613 remaining papers we reject, they match at least one of the criteria described in Table 2.

We used the Cohen Kappa method to improve the quality of the results and measure the level of agreement among the researchers. To do so, two researchers analyzed separately a sample containing 140 (10%) of the 1314 papers. We performed four reviews iterations, having 35 papers double-analyzed in each iteration. After each cycle, we discussed the papers where we had a different opinion about accepting or rejecting until we reach an agreement. Table 3 presents Kappa values. At the end of the fourth review cycle, we reached an almost perfect agreement, which suggests that researchers are aligned and can follow screening the papers independently.

We use StArt (State of the Art through systematic review)[2] to assist us in the data extraction phase. After completing data extraction using StArt, we export the results to a Google Sheets, so that we can collaboratively analyze the data. At the end of this process, we accept a set containing 177 accepted articles, which are in our Technical Report (TR) (Borges et al., 2020).

3 RESULTS

In this section, we present how we classify the 177 papers we accepted, and also the data we extracted from the papers. We applied the search string in July/2019 on the following web search engines: ACM, IEEE, Springer, arXiv, Science Direct, Web of Science, Google Scholar, and Scopus. We do not limit the years to retrieve the papers because we want to understand how this area developed since the beginning. Table 4 presents the amount of papers per database. Scopus is the search engine that returned most papers because it indexes multiple databases, although

[2] Available at http://lapes.dc.ufscar.br/tools/start_tool

more than half the papers we accepted are from IEEE and ACM. We have to mention that regarding the presented percentages of domains, techniques, and tools, some papers presented more than one of the items, so the sum of the percentage is always greater than 100%, for all the answers.

3.1 Data Extraction

We thoroughly read the 177 papers so that we could analyze and classify them. We retrieved 113 journals and 64 papers published in conferences/events. The oldest ones are from 1992 and 1993, written by the same authors, Lionel C. Briand, Victor Basili, C. J. Hetmanski, and W. Thomas. The papers [R29, R28, R30] use a Decision Tree-based approach to Optimized Set Reduction (OSR) for SE data analysis. These papers perform software cost estimation and failure classification in software projects and software components. Since then, the number of related papers has remained stable until 2008. In 2009, there was a growth in publications related to the use of ML techniques in SE, reaching a peak of 42 papers in the year 2018. This scenario shows that the use of ML techniques in SE is feasible and quite useful for solving significant problems in the area due to its diversity of available applications.

3.2 Classification Schema

We created a schema to present our results. We divide the schema according to each research question. We now present the resulting analysis and how we answer each research question.

RQ1: How has Machine Learning been used in Software Engineering? Using ML, we create two hierarchical clusters (Figure 1), setting Ward as the Linkage parameter. A hierarchical clustering diagram is a form of visualization that unifies the most related to the least related items. The weighted link used by the Orange tool to create hierarchical clustering is the WPGMA (Weighted Pair Group Method with Averaging) method, as described in (Dubes and Jain, 1988). In this paper, the diagram presents the grouping of the most related papers. Initially, it unifies in pairs that relate more strongly and from this union, will be united with the other papers. In one example, papers R80 and R58 initiate a cluster within Cluster 01. Both papers deal with software effort estimation. From this union, the remaining papers are grouped by similarity of this context. Below we describe the clusters.

A) Cluster 1 - 122 Papers
Most frequent terms: *model* (248), *prediction* (221), *project* (206), *data* (197), *technique* (187), *based* (138), and *result* (136).

Table 4: Papers per electronic databases.

Source	Initial	Selection	Accepted
Scopus	395	30	17 papers
IEEE Xplore	307	115	62 papers
Web of Science	283	61	35 papers
ACM	158	66	32 papers
Springer	69	14	5 papers
arXiv	42	18	7 papers
Science Direct	42	21	19 papers
Google Scholar	18	3	0 papers
	1314	328	177 papers

Most papers in Cluster 1 are related to the Software Quality KA (66 papers) and Software Engineering Management (40 papers). The most mentioned context in these papers is defect/bug prediction (25 papers), and the ML technique most used is LR (40 papers). WEKA is the most used tool (38 papers).

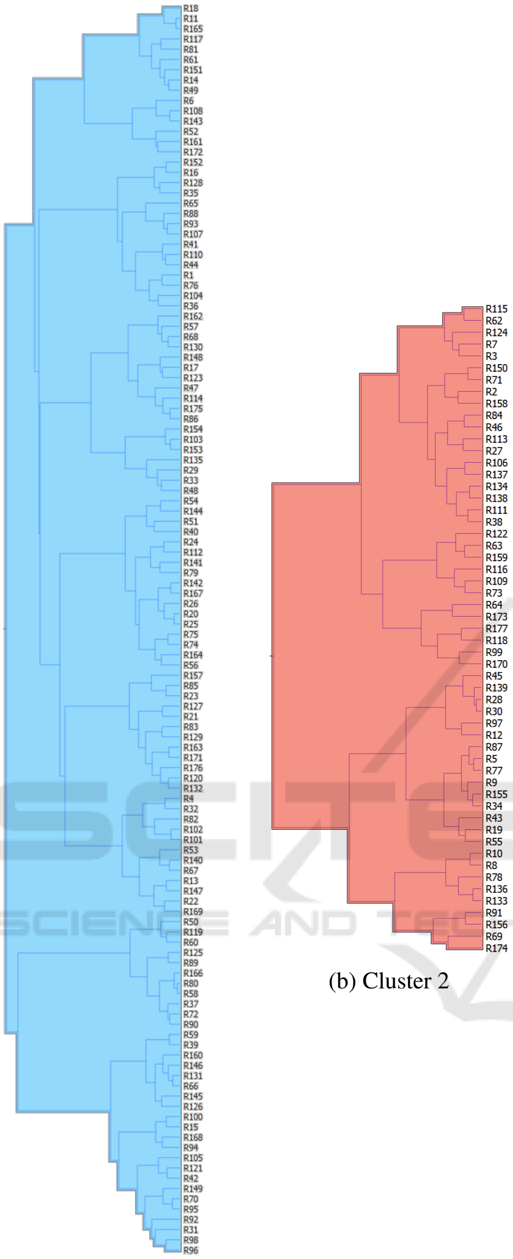
B) Cluster 2 - 55 Papers
Most frequent terms: *approach* (114), *based* (73), *system* (73), *model* (47), *technique* (45), *analysis* (42), and *problem* (39).

Papers in Cluster 2 are mainly about Software Quality (25 papers), and Software Test (15 papers). Once again, WEKA is the most frequently used tool (18 papers). NB is the most used ML technique (16 papers) to solve problems in the defect classification context (6 papers).

Cluster 1 contains the papers most related to prediction, while Cluster 2 has the most related to classification. We found that the word *predict* appears 392 times, considering all 177 papers. It suggests that most papers work to address problems in SE using prediction techniques. The word *defect* appears 199 times, which is consistent with software defect being the most common reported domain issue.

In the second paper classification approach we adopted, we link the ML tools and techniques to the SE domain. To do so, we use the ten SWEBOK KA presented in Table 1. We present the results in descending order, according to the number of papers:

- **Software Quality:** 91 papers: [R1, R3, R4, R7, R11, R19-R21, R23-R25, R28, R30, R32, R35, R36, R40, R45, R48-R54, R56, R57, R61-R65, R68, R70, R71, R74-R78, R80-R86, R89, R90, R91, R96, R97, R99, R101, R102, R104, R107, R108, R110, R112, R115, R116, R120, R121, R125, R127-R132, R136-R138, R144, R146, R148, R149, R151, R156, R157, R159, R161, R164-R166, R169-R171, R174, R176].
- **Software Engineering Management:** 46 papers:



(b) Cluster 2

(a) Cluster 1

Figure 1: Clustering of papers.

[R2, R10, R13, R15-R17, R20, R29, R31, R36, R39-R42, R46, R47, R55, R60, R66, R67, R72, R83, R89, R92-R94, R96, R98, R103, R112, R114, R122, R123, R126, R129, R140, R141, R143, R147, R152-R155, R162, R163, R175].

- **Software Testing:** 28 papers: [R5, R6, R19, R24, R27, R28, R40, R43, R58, R61, R63, R69, R78, R87, R91, R100, R105, R134, R135, R137-R139, R157, R160, R165, R168, R171, R177].
- **Software Engineering Models and Methods:** 22 papers: [R6, R22, R28, R42, R55, R62, R69, R72, R73, R106, R108, R110, R111, R117, R118, R120, R139, R152, R165, R170, R172, R175].
- **Software Configuration Management:** 15 papers: [R2, R12, R37, R44, R54, R70, R111, R113, R124, R142, R158, R167, R172, R173, R18].
- **Software Maintenance:** 13 papers [R8, R33, R50, R51, R52, R57, R81, R88, R102, R115, R139, R166, R170].
- **Software Requirements:** 8 papers: [R9, R41, R79, R109, R114, R117, R121, R150].
- **Software Engineering Process:** 8 papers: [R14, R38, R46, R90, R95, R123, R145, R169].
- **Software Design:** 6 papers [R34, R40, R79, R117-R119].
- **Software Construction:** 2 papers [R76, R167].

During the classification process, we identify that more than half of the papers discuss the use of ML for Software Quality (52%). The second KA most benefited from ML is Software Engineering Management (26%), followed by Software Testing (16%).

RQ2: Which Software Engineering Domain Issues can Benefit most from using Machine Learning? We analyze the papers and identify 86 different domains. The 8 most frequent domains were: Software Defect (56 papers), Software Effort (16 papers), Bug Report (6 papers), Risk Management (5 papers), Software Cost (4 papers), Software Requirements (4 papers), Source Code Refactoring (4 papers), and Change Prone Parts (4 papers). We highlight the three most frequent ones:

- **Software Defect (31%):** the papers we classify in the software defect domain present the use of ML for bug classification, prediction, and retrieval. Papers: [R3, R7, R10, R11, R20, R21, R23, R24, R28, R35, R49, R50, R53, R54, R57, R59, R61, R63, R64, R66, R74, R75, R78, R80, R82, R83, R84, R85, R86, R89, R91, R99-R101, R104, R107, R110, R112, R120, R125, R127, R128, R130, R132, R136, R137, R138, R144, R149, R151, R157, R161, R165, R168, R171, and R176].
- **Software Effort (9%):** ML tools and techniques used to predict software effort, helping in software project management. Papers: [R13, R16, R20, R67, R92, R96, R103, R140, R141, R143, R147, R153, R154, R162, R163, and R175].
- **Bug Report (3%):** In the bug report domain, they use ML for bug deduplication. Papers: [R1, R5, R45, R148, R146, and R156].

RQ3: What Computational Machine Learning Techniques and Tools are Most used to Solve Software Engineering Problem Domains? From the

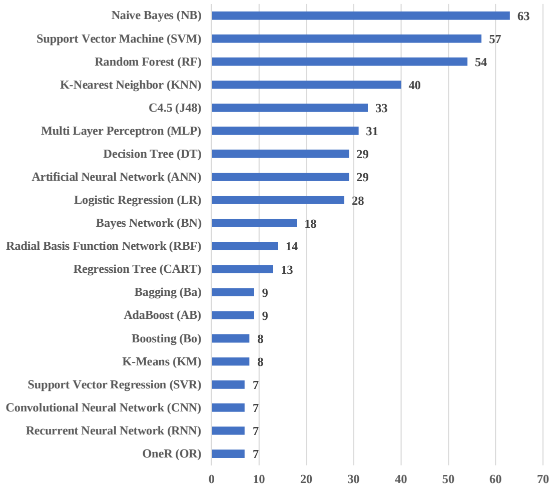


Figure 2: 20 most frequent ML techniques used in SE domain.

problem domains presented, we identified 328 ML techniques and 35 tools used to assist in solving SE problems. We list ML techniques, as described in the papers. Even though some techniques such as neural networks (ANN, CNN, and RNN) and tree classification algorithms (C4.5 (J48), DT, and CART) could be grouped, we choose to keep the original descriptions we found in the papers, to know exactly the most used algorithms. Figure 2 presents the 20 most used techniques and Figure 3 illustrates the 12 most used tools. Below we list the most frequently mentioned techniques and tools.

- **Techniques/Methods:**
  1. **Naïve Bayes (Rish et al., 2001) (36% of Papers):** NB is a classifier that keeps statistics about each column of data in each class. New examples are classified by a statistical analysis that reports the closest class to the test case. NB classifiers are computationally efficient and tend to perform well on relatively small datasets. Papers: [R1, R2, R3, R5, R7, R11, R13, R14, R21, R23, R24, R31, R36-R38, R56, R57, R60, R64, R66, R73, R74, R75, R77, R82, R83, R88, R91, R97, R99, R100, R102, R105, R107, R110, R111, R115-R120, R122, R126, R128, R129, R131, R132, R142, R145-R150, R152, R154, R155, R161, R162, R164, R168, and R172].
  2. **Random Forest (Liaw et al., 2002) (30% of Papers):** RF is a joint learning process that combines multiple weaker learners into a stronger learner. RF have a better generalization and are less susceptible to overfitting (Breiman, 2001). They can be used for classification and regression problems. Papers: [R3, R4, R11, R13, R21, R31, R37, R39, R49, R51, R54, R56, R57, R59, R60, R61, R62, R65, R73, R75, R80, R82, R83, R88, R99, R100, R102, R104, R105, R109, R110, R112, R115-R119, R122, R123, R127, R128, R129, R131, R132, R134, R140, R145, R148-R150, R152, R159, and R171].
  3. **Support Vector Machine (Cristianini et al., 2000) (32% of Papers):** It is responsible for creating a linear discrimination function using a small number of critical threshold instances (called support vectors) of each class, ensuring maximum possible separation (Liu et al., 2002). SVM is known as SMO in the WEKA tool. Papers: [R1, R2, R4, R7, R11, R16, R22, R24, R31, R32, R38, R42, R51, R57, R59, R60, R61, R62, R63, R66, R67, R68, R70, R73, R74, R77, R80, R82, R83, R84, R86, R89, R90, R91, R94, R105, R108, R110, R111, R115, R119, R120, R122, R124, R125, R126, R128, R143, R145, R146, R161, R162, R164, R165, R171, R172, R174].
- **Tools:**
  1. **WEKA (32% of Papers) (Hall et al., 2009):** WEKA is a unified work environment that allows researchers access to ML techniques previously implemented and easily configured. Besides providing a learning algorithm toolbox, WEKA also provides a framework so researchers can implement new algorithms without having to worry about infrastructure support for data manipulation and schema evaluation. Papers: [R1- R5, R7, R10, R14, R17, R19, R21, R23, R24, R27, R31, R34, R36, R47, R49, R51, R53, R54, R56, R57, R64, R67, R75, R77, R82-R86, R99-R101, R103, R109, R110, R117-R119, R121, R124, R128, R129, R131, R134, R135, R142, R145, R152, R154, R155, R161, R164, and R177].
  2. **MATLAB (9% of Papers)[3]:** Platform to solve scientific and engineering problems. The matrix-based MATLAB language serves to express computational mathematics. It is used for ML, signal processing, image processing, machine vision, communications, computer finance, control design, robotics, and many other fields. Papers: [R2, R3, R8, R35, R48, R55, R81, R85, R92, R93, R127, R140, R141, R144, R149, and R160].
  3. **SCIKIT-LEARN[4] (8% of Papers):** A Python module that integrates a wide range of ML algorithms for medium-scale supervised and un-

[3] Available at: http://la.mathworks.com/

[4] Available at: https://scikit-learn.org/stable

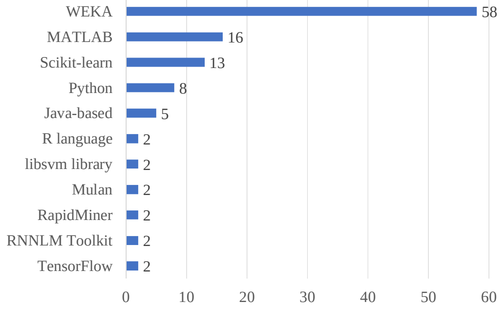


Figure 3: 11 most frequent ML tools used in SE domain.

supervised problems. This package focuses on bringing ML to non-specialists using a high-level language for general use. Papers: [R63, R65, R69, R73, R80, R104, R105, R115, R116, R122, R148, R150, and R168].

On the most used tools, we found 19 papers (11%) reporting the programming language Python or Python-based Scikit-learn, libsvm, and TensorFlow (R45, R51, R63, R65, R69, R73, R80, R94, R104, R105, R115, R116, R122, R148, R150, R157, R166, R168, R170).

4 CONCLUSION

In this paper, we presented a mapping study developed to understand how SE benefits from using ML tools and techniques. In the final set, we accepted 177 papers that answer our research questions.

During papers' analysis, we found more than three hundred ML techniques, although only ten are present in more than 10% of the papers: NB, RF, SVM, KNN, C4.5, DT, ANN, LR, and BN. NB is used in 36% of papers, and it is because the algorithm performs well tasks related to text classification. NB is widely used in SE to classify data related to source code, bugs, and reports. All top 10 algorithms are classifiers or classifiers/regression, and they are in the supervised learning paradigm. Thence, we noticed there is an opportunity to expand the exploration of unsupervised and reinforcement learning techniques for the SE domain.

We identify 35 different ML tools in 111 papers, being WEKA the most used. According to a survey performed by Kaggle[5], Python is the most recommended programming language for data science (75% of respondents). In our study, we found Python or Python-based tools in 11% of papers. As 67 papers do not mention the used tools, this number could be bigger. The papers that present ML techniques (37%), but not the tools, libraries, and frameworks make it difficult to replicate the experiments presented, as well as to understand the environment configuration.

We also cluster the papers using ML to identify and rank the most similar papers in terms of content. We identify two hierarchical clusters, one focused on prediction (69%) and the other on classification (31%) tasks. The word *predict* is the most frequent in all papers, which leads us to conclude that ML is mainly used to make predictions in SE. We also found that Software Quality is the most frequent SWEBOK KA target for both clusters (51%).

ML provides a set of advantageous features for software quality assessment and prediction. One of the most common ways to analyze quality is through the use of software metrics. They assist in decision making and serve as input to ML models to propose predictions or ratings about software quality. In this sense, we find that DT, NN, NB, and Classification and Regression Tree algorithms are most used for creating models for design artifacts prediction. Quality analysts use these models to measure design artifacts to assist in quality assessment early in the development process. With these models, it is possible to generate a more objective and automated project quality assessment. Project metrics are also used in quality prediction, generating models from DT, RF, LR, and NB algorithms. However, to speed up the models, the input metrics need to be specific to a particular project context, which is the case of the prediction of the failure propensity of specific modules of the software. As a result of this type of forecasting, the test analyst can predict software quality even before implementation.

Being a qualitative research, our paper has some limitations. To minimize the risk of not having a broad and diverse range of documents, we conducted our research on eight popular web search engines. However, this is not a guarantee that all events and magazines in the area are covered. The fact that papers that work on related topics appear in different locations may lead to different results. In the selection phase, we did not delimit an initial year for the papers, so it could bring tools and techniques that are no longer being used. Since we wanted to map ML to SE from the beginning, it was an acceptable limitation, but to ensure exciting results, we also looked at the algorithms described in new papers since 1992 as OSR, and are still used in recent papers. We apply the Kappa method to improve agreement and reduce interpretability bias. Two researchers worked in parallel, and we had two other researchers to discuss in case of disagreement.

For future work, we want to map the challenges

[5] Available at https://www.kaggle.com/kaggle/kaggle-survey-2018

related to using ML in SE projects. We also suggest exploring the use of ML in areas that are little explored, such as Software Construction (1%), Software Design (3%), Software Requirements (5%), and Software Engineering Process (5%). Additionally, we suggest there is an opportunity to increase the application of reinforcement and unsupervised learning for SE tasks.

ACKNOWLEDGEMENTS

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

REFERENCES

Al-Nusirat, A., Hanandeh, F., Kharabsheh, M., Al-Ayyoub, M., and Al-dhufairi, N. (2019). Dynamic detection of software defects using supervised learning techniques. *Int. Journ. of Comm. Net. and Info. Secy.*, 11:185–191.

Anu, V., Hu, W., Carver, J. C., Walia, G. S., and Bradshaw, G. (2018). Development of a human error taxonomy for software requirements: a systematic literature review. *Info. and Soft. Technol.*, 103:112–124.

Borges, O., Couto, J., Ruiz, D., and Prikladnicki, R. (2020). Technical report - how machine learning has been applied in software engineering? https://figshare.com/articles/TR-MUNDDOS-2020_ICEIS_Olimar_pdf/11874231.

Bourque, P., Fairley, R. E., et al. (2014). *Guide to the software engineering body of knowledge (SWEBOK (R)): Version 3.0*. IEEE.

Breiman, L. (2001). Random forests, machine learning 45. *Journ. of Clinical Microbiology*, 2:199–228.

Choetkiertikul, M., Dam, H. K., Tran, T., Pham, T. T. M., Ghose, A., and Menzies, T. (2018). A deep learning model for estimating story points. *IEEE Trans. on Soft. Eng.*, 45:637–656.

Cristianini, N., Shawe-Taylor, J., et al. (2000). *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press.

De Rezende, L. B., Blackwell, P., and Pessanha Goncalves, M. D. (2018). Research focuses, trends, and major findings on project complexity: a bibliometric network analysis of 50 years of project complexity research. *Proj. Mgmt. Journ.*, 49:42–56.

Duarte, C. H. C. (2019). The quest for productivity in software engineering: A practitioners systematic literature review. In *Int. Conf. on Soft. and Syst. Proc.*, pages 145–154. ACM.

Dubes, R. C. and Jain, A. K. (1988). Algorithms for clustering data.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. (2009). The weka data mining software: an update. *SIGKDD Explor. Newsletter*, 11:10–18.

ISO/IEC/IEE (2010). Iso/iec/ieee international standard - systems and software engineering – vocabulary. *ISO/IEC/IEEE 24765:2010(E)*, pages 1–418.

Kitchenham, B. and Charters, S. (2007). Guidelines for performing systematic literature reviews in software engineering. Technical Report EBSE-2007-01, Dep. Comp. Sci, University of Durham, Durham, UK.

Kumar, L., Satapathy, S. M., and Murthy, L. B. (2019). Method level refactoring prediction on 5 open source java projects using machine learning techniques. In *Innov. on Soft. Eng. Conf.*, page 7. ACM.

Landis, J. R. and Koch, G. G. (1977). The measurement of observer agreement for categorical data. *Biometrics*, 33:159–174.

Liaw, A., Wiener, M., et al. (2002). Classification and regression by randomforest. *R news*, 2:18–22.

Liu, B.-B., Dong, W., and Wang, J. (2018). Survey on intelligent search and construction methods of program. *Journ. of Soft.*, 29:2180–2197.

Liu, H., Li, J., and Wong, L. (2002). A comparative study on feature selection and classification methods using gene expression profiles and proteomic patterns. *Genome Inform.*, 13:51–60.

Mitchell, T. (1997). *Machine Learning*. McGraw-Hill.

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

Petersen, K., Vakkalanka, S., and Kuzniarz, L. (2015). Guidelines for conducting systematic mapping studies in software engineering: An update. *Info. and Soft.Technol.*, 64:1 – 18.

Rish, I. et al. (2001). An empirical study of the naive bayes classifier. In *Wksh. on Emp. Meth. in AI*, pages 41–46. CiteSeer.

Sedano, T., Ralph, P., and Péraire, C. (2017). Software development waste. In *Int. Conf. on Soft. Eng.*, pages 130–140. IEEE.

Zhang, D. and Tsai, J. J. (2003). Machine learning and software engineering. *Soft. Qual. Journ.*, 11:87–119.