Systematic Mapping of Machine Learning in Software Engineering

Aris J. Aristorenas
University of Calgary
Calgary, AB, CA
ajaristo@ucalgary.ca

ABSTRACT

This midterm report discusses the progress of the machine learning systematic mapping study in software data analytics. The motivation of the study is first discussed, followed by a presentation of the selected research questions, along with their rationale. Next details are presented regarding the initial scoping and validation, as well as the systematic steps taken so far. One of these steps is with the iterative process to continuously refine keyword searches - a crucial part of the systematic mapping. Finally, the results of the initial classification of the 40 papers are presented.

KEYWORDS

software engineering, machine learning, data analytics

ACM Reference Format:
Aris J. Aristorenas. 2021. Systematic Mapping of Machine Learning in Software Engineering. In . ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 MOTIVATION

Machine Learning (ML) is a branch of artificial intelligence which aims to find patterns within a data set, and make decisions using that data set as an input, with minimal human intervention. The theoretical formulation of machine learning has several statistical underpinnings but these days programming libraries like SciKit-Learn, and statistical packages like R, can handle much of the computation, allowing the user to almost consume the ML algorithms as a blackbox. Machine Learning technology has lead to breakthroughs in several domains: automating spam detection in emails (cybersecurity), predicting the stock market (finance), detecting malignant cells and anomalies in medical scans (medicine), and self-driving vehicles (automative). These breakthroughs have contributed to the amount of research that has been done within the past decade (2010-Present).

Although the accessibility of machine learning has increased through libraries like SciKit-Learn, where one can build, run, and test their own machine learning model in just a few lines of code, there are several dangers in place in consuming ML in this blackbox way. First is with failing to reconcile the machine learning results with domain knowledge experts in the field of where the data resides: machine learning will always output a result, whether or not it makes sense in the original context falls within the due diligence of the data science team to qualify with the help of domain experts.

With regards to the machine learning process (or pipeline) itself, it is beneficial to know how researchers are carrying out their studies: is there a common ML algorithm used, and if so why? What steps of the ML pipeline are followed, and in the evaluation of a ML model, what accuracy metrics are presented. Answer to these questions are beneficial to know as a feed-in to future studies that extend this systematic mapping (such as a SLR), but are also useful to know when observing the mapping results holistically for any correlations. As an example question is: "do researchers only use the random forest algorithm for a certain ML problem in software engineering (like predicting defects)", or: "do researchers only use the F1 score for a specific software use-case?" The results of this systematic mapping study may reveal any potential correlations like this.

Lastly, on the study of ML return-on-investment (ROI), which is a metric that quantifies the additional benefit that can be obtained through an incremental increase in model accuracy, several of the research questions of this systematic mapping study may help to further drive this research area.

2 RESEARCH QUESTIONS

Below are the research questions for this systematic mapping, along with a rationale for each.

RQ1. What studies have reported on accuracy results such as F1, and AUC, and if neither, what other metrics did that study use to report their model accuracy?

RQ2. What studies present evidence quantifying the impact of the machine learning results and benefit in the original domain-context?

RQ3. How many studies present the the entire machine learning pipeline, and for those that do not discuss the entire pipeline, which elements of the pipeline were discussed?

RQ4. How many studies consider multiple ML algorithms, and what is the distribution of algorithms out of all the studies sampled?

At the core of a successful machine learning model is the ability of that model to accurately make predictions on unseen data. Typically, this is quantified by one or more metrics such as accuracy, precision, recall, specificity, and F1 score. The rationale of the first research question (RQ1) is to produce some snapshot of the status quo as to what metrics researchers currently use to quantify model accuracy. Insight into this may potentially lead to uncover

University of Calgary '21, July - Aug, 2021, Calgary, AB

ACM ISBN XXX-X-XXXX-XXXX-X/XX/XX…$XX.XX
https://doi.org/10.1145/nnnnnnn.nnnnnnn

some correlations between accuracy metrics and machine learning use-cases in software engineering.

Also at the core of a successful machine learning model is its ability to demonstrate its merit in the original context of the domain problem. There have been machine learning studies which present all of the results including the accuracy of the model, and end there without providing any details about the impact, results, and benefits of applying the model towards the original domain problem. An example of a study in medicine would be developing a model which can detect malignant cells in MRI scans, going through the details of the machine learning pipeline, and reporting on the accuracy on some test set, but then omitting details about any real-world application on the model analogous to how clinical trials are done when a new vaccine is developed: one would not just develop the vaccine and stop there, they would test its overall merit and impact. RQ2 aims to answer what proportion of studies quantify the real benefits of their machine learning model beyond model accuracy.

RQ3 aims to uncover a frequency distribution of the elements of a machine learning pipeline that a study presents. The main elements are with data collection, data pre-processing, model train and test, model evaluation, and model deployment. A hypothesis is that the vast majority of time spent in the machine learning pipeline is spent with data pre-processing. Indeed a machine learning model can only be as good as the data that is fed as an input, so it makes sense to invest in a good pre-processing layer. Although it is difficult to gauge a percentage number in this systematic mapping as to how much time a study has spent on a certain element, RQ3 will reveal what the status-quo is in terms of commonalities between what elements of the pipeline steps studies present. This is important information to know as a precursor to quantifying on a finer level of granularity, the time spent in each step of a machine learning pipeline (as a potential future study).

Lastly, RQ4 aims to understand what the frequency distribution is by machine learning algorithms in the studies that fall within the scope of this systematic mapping. These algorithms include linear regression, logistic regression, decision trees, random forests, hierarchical clustering, support vector machines, and various others. The answer to this RQ4 will shed light on the status quo of what algorithms researchers use in their ML problems, and may potentially also reveal correlations: in software bug detection for example, do researchers always use logistic regression, or are other algorithms preferred?

3 SCOPING AND INITIAL VALIDATION

In this section, the steps taken so far in the systematic workflow are presented. The initial steps follow the process outlined by Petersen et al., with the goal being to structure the research area of machine learning in software engineering and software development. As of this midterm report submission, the steps so far have been:

(1) development of keywords,
(2) development of the RQs,
(3) initial scoping and retrieval of papers using the keywords in the selected databases,
(4) first classification attempt
(5) development of exclusion criteria
(6) aggregating the results for a first assessment,
(7) refinement of the keywords and RQs, based on this assessment,
(8) second pass of paper retrieval and classification, and

Among the very first steps taken was to develop the keywords to be used in journal database searches. In this first pass, the keywords were "software engineering" AND "machine learning" AND "data analytics". Details about the iterations and refinements to keyword searches are described in section 4. Next, a first pass was done to develop the research questions. Those RQs presented in section 2 were not the original RQ set: the first pass consisted in RQs that were either difficult to classify, or were not formulated so as to produce some categorical binning in the eventual answer to that RQ. An example of an original RQ of the former was: "What proportion-of-time distribution of data preparation activities (data engineering, feature engineering, and data cleansing) in machine learning in studies related to software engineering?" with the categories as 0-10%, 10-20%, 20-30%, and so on. Although these are legitimate categories, this would prove to be non-feasible for the systematic mapping of this study: many papers do not formally state what proportion of time their entire study devoted to data pre-processing, for example. Therefore, the the original RQ set was refined (to the set in Section 2) to be able to apply some classification scheme in a more mechanical process (the process is described in section 4, with an example workflow presented in section 6). Initially, the refined set involved categorical bins with simple "yes/no" categories, but were later refined to be more detailed, for instance instead of "yes/no" to the accuracy in RQ1, the refinement involved listing out the accuracy metrics reported by a study.

With the RQs in place, and an initial formulation of the keywords, a search was done in IEEE Xplore resulting in 331 papers (259 conference, 51 journal, and 21 articles/magazines) as step 3. In step 4, the classification was then performed in a subsample of the first 23 search results. In this sample, an exclusion criteria became easier to develop, which was step 5 of the systematic mapping work done so far. From RQ1, and RQ3, papers that were theory-based were excluded as these studies typically did not present on any machine learning pipeline. An example is in Lu, et al. [2]. The title of this study contains "Lie Group", a term coming from pure mathematics, and upon examining the paper itself, although the paper was on machine learning, it was theory-based with no focus on any ML pipelines. More examples of the classification and refinement to the exclusion criteria are presented in section 6.

In step 6, after the classification was applied to the 23 papers, the results were aggregated into histogram visualizations. The purpose of this was to quality check the the RQs, and keywords to see if there were any imbalances. For example, at this stage, if one of the yes/no categories contained 22 papers which answered yes, the RQs might need some tuning and refinement. Although the results appeared to be balanced, the RQs and keywords were still refined, which is what was done in step 7.

This first-pass revealed several potential improvements: first, the categories of "yes or no" could be further subdivided to allow for more a more detailed frequency distribution. This was applied to RQ1, and RQ3. Originally, these questions were:

(1) Paper exclusively provides accurate results like F1, AUC: Yes or No

(2) Paper looks at the whole process of running the analytics: Yes, or No, or Partially

Because it is relatively quick to identify the accuracy metrics, and elements of a ML pipeline a study presents, these two RQs were refined to those in section 2: with some additional work, instead of answering just "yes or no", the actual metrics used by that study were instead recorded (similar with the steps of the ML pipeline). Within this step is also where RQ4 was added.

4 KEYWORD SEARCH AND REFINEMENTS

The initial keyword search was refined to include multiple facets of general software, not just software engineering. These included: Software Development, Software Engineering, and Software Lifecycle. At this stage, another inclusion criteria identified was also used to refine the keyword search. A vast majority of studies that were on the subject of big data, which met the keyword search criteria, had very little to do with the machine learning process, and were more geared towards cloud architecture solutions for big data. As a result, a "NOT Big Data" was appended to the keyword search string. Details of how the number of papers that were returned changed are presented in Section 5. With the refined keyword search, and RQs, a second pass was done to retrieve another collection of studies, which brought the subsample to 40. The classification (using the refined RQs) was then performed on these 40 papers. In doing so, further refinements were developed: a handful of papers were too focused in a particular domain, such as finance, geophysics, or astrophysics. These made it through the search because each paper likely has the keyword "software" in it, as well as "machine learning". Another refinement was done to append NOT "Geophysics", NOT "Astrophysics", NOT "Finance" to the search string.

5 SEARCH RESULTS

Below in Figure 1 are the results of the search in the IEEE Xplore database using the initial keywords. In this search, as mentioned previously, 331 papers resulted including 259 conference papers, 51 journal papers, and the rest were papers that can be omitted (early access articles, magazines, and standards).

Figure 1: IEEE Xplore search result of initial keywords

The first refinement to this search was to account for other areas of software, such as the software lifecycle, software development, and software analytics. This is shown in Figure 2. As shown, the search resulted in 610 papers including 509 conference papers, and 67 journal papers.

The second refinement to the search was to exclude papers on big data. This is shown in Figure 3.

This resulted in 346 papers, of which 291 were conference papers, and 32 were journal papers. Since one of the criteria of this systematic mapping study is to consider at least 3 databases, and in total, include under 200 papers, this number was still too high. Further refinements were necessary. One of those was to only include papers from 2010 to the present. The result is shown in Figure 4:

Figure 2: IEEE Xplore search result of first keyword refinement

Figure 3: IEEE Xplore search result of second keyword refinement

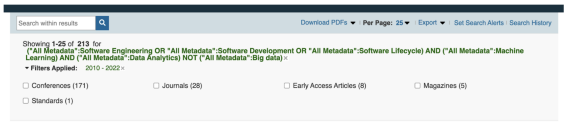


Figure 4: IEEE Xplore search result of third refinement

The results were now at 213, with 171 papers coming from conferences, and 28 coming from journal papers. The last refinement made in this process was based on improving the search results which could help answer RQ2. Here, keywords "Experimental", and "Empirical" were added with an AND operator, as shown in Figure 5.

Figure 5: IEEE Xplore search result of fourth refinement

This resulted in 35 papers, including 23 conference papers, and 9 journal papers. This number is more ideal because if similar results were obtained in the 2 other databases, the cumulative total would be under 200. As of this midterm report however, Scopus is returning several hundred papers with the same keyword search, some troubleshooting is still being done to reduce this number without significant changes to the keyword search (to keep the search string constant throughout the 3 databases).

6 VALIDATION

In this final section, the systematic mapping classification scheme is presented, and applied to a selected subsample of the 40 papers that have been classified. A Google Doc, and Google Sheet is used for raw data collection. The steps applied to each paper are as follows:

(1) Review the abstract of the paper, and check if any of the RQs can be directly answered.

(2) In reviewing the abstract of the paper, check if any of the exclusion criteria are met. If one or more are met, exclude the paper, but also validate this decision by quickly scanning the body of the paper.
(3) If some of the RQs can be answered in the abstract, but not all, search for any methodology section in the body where the study talks about the machine learning process they followed. If one is found, then identify the steps of the machine learning pipeline that were followed, and record this for answering RQ3 in the Google Doc
(4) Check the methodology, and results section of the paper to see if the paper discusses which machine learning algorithms were used in their study. Record any of these algorithms in the Google Doc to help answer RQ4.
(5) Check the evaluation or validation section of the paper to see what accuracy metrics the paper reports. Record any of these metrics in the Google Doc to answer RQ1.
(6) If there is no section where the paper presents a machine learning pipeline, try searching the document using keywords from the RQs, such as "accuracy", "validation", and "algorithm". It's possible that the paper reports on accuracy, but in short paragraph of the paper that is easy to miss at first glance. If this is the case, record any metrics, ML algorithms, or ML pipeline steps that the paper follows in the Google Doc.
(7) If after this document search, there is no accuracy, nor any ML algorithms, then it is possible that the paper falls under the exclusion criteria of "Paper does not conduct a formal ML study". The "empirical", and "experimental" keywords should have excluded these, but it's possible that the paper still made it through. In this case, the review the body of the paper to validate the decision to potentially exclude it from the study. Record the citation, in the Google Doc, and list the reason for the exclusion.
(8) Finally, to answer RQ2, review both the results, and discussion section to identify if the paper presents any real-world application of their ML model to answer the domain problem in the original context, as well as its impact. If there is any evidence which quantifies this impact, record an answer of "yes" to RQ2, otherwise record "no".
(9) Repeat the above steps for each paper

Following now is an application of this scheme in a 4 selected papers. The first paper is by Assim, et al. (2020) [1]. The abstract of this paper is shown for conveniences purposes in Figure 6 to apply the above scheme.

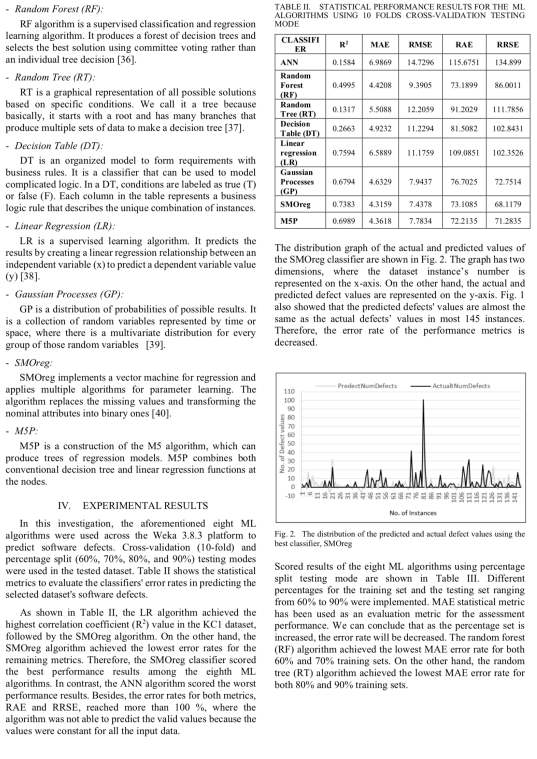
As seen from this abstract, we can immediately identify a variety of ML algorithms that were used, and the paper clearly has a focus of using ML in the context of some software engineering or software development area, in this case, predicting software defects. Further examination of the paper body is warranted. Scanning for sections such as the methodology section reveals that the the paper explicitly lists all the ML algorithms that were used. This is depicted in Figure 7.

These ML algorithms are recorded in the Google Doc (Step 4 of the classification scheme). Also appearing in this Figure is a table of statistical performance metrics, in this case: R^2, MAE, RMSE, RAE, and RRSE. These are the answers to RQ1, and are recorded into the Google Doc. Another scan of the paper's methodology section shows that training, validation, prediction, and evaluation were the steps of the ML pipeline that were followed. There was in fact no

***Abstract*— Software development and the maintenance life cycle are lengthy processes. However, the possibility of having defects in the software can be high. Software reliability and performance are essential measures of software success, which affects user satisfaction and software cost. Predicting software defects using machine learning (ML) algorithms is one approach in this direction. Implementing this approach in the earlier stages of the software development improves software performance quality and reduces software maintenance cost. Different models and techniques have been implemented in many studies to predict software defects. This investigation implements ML algorithms, such as artificial neural networks (ANNs), random forest (RF), random tree (RT), decision table (DT), linear regression (LR), gaussian processes (GP), SMOreg, and M5P. A new software defect prediction model for software future defect prediction is proposed. The defect prediction is based on historical data. The results showed that a combination of ML algorithms could be used effectively to predict software defects. The SMOreg classifier scored the best performance results, where the ANN classifier scores the worst results.**

***Keywords*— software defect prediction, software defect, prediction model, machine learning (ML), artificial neural networks (ANN), SMOreg Classifier.**

Figure 6: Abstract of Assim, et al. (2020) in [1]



- *Random Forest (RF):*

RF algorithm is a supervised classification and regression learning algorithm. It produces a forest of decision trees and selects the best solution using committee voting rather than an individual tree decision [36].

- *Random Tree (RT):*

RT is a graphical representation of all possible solutions based on specific conditions. We call it a tree because basically, it starts with a root and has many branches that produce multiple sets of data to make a decision tree [37].

- *Decision Table (DT):*

DT is an organized model to form requirements with business rules. It is a classifier that can be used to model complicated logic. In a DT, conditions are labeled as true (T) or false (F). Each column in the table represents a business logic rule that describes the unique combination of instances.

- *Linear Regression (LR):*

LR is a supervised learning algorithm. It predicts the results by creating a linear regression relationship between an independent variable (x) to predict a dependent variable value (y) [38].

- *Gaussian Processes (GP):*

GP is a distribution of probabilities of possible results. It is a collection of random variables represented by time or space, where there is a multivariate distribution for every group of those random variables [39].

- *SMOreg:*

SMOreg implements a vector machine for regression and applies multiple algorithms for parameter learning. The algorithm replaces the missing values and transforming the nominal attributes into binary ones [40].

- *M5P:*

M5P is a construction of the M5 algorithm, which can produce trees of regression models. M5P combines both conventional decision tree and linear regression functions at the nodes.

IV. EXPERIMENTAL RESULTS

In this investigation, the aforementioned eight ML algorithms were used across the Weka 3.8.3 platform to predict software defects. Cross-validation (10-fold) and percentage split (60%, 70%, 80%, and 90%) testing modes were used in the tested dataset. Table II shows the statistical metrics to evaluate the classifiers' error rates in predicting the selected dataset's software defects.

As shown in Table II, the LR algorithm achieved the highest correlation coefficient (R^2) value in the KC1 dataset, followed by the SMOreg algorithm. On the other hand, the SMOreg algorithm achieved the lowest error rates for the remaining metrics. Therefore, the SMOreg classifier scored the best performance results among the eighth ML algorithms. In contrast, the ANN algorithm scored the worst performance results. Besides, the error rates for both metrics, RAE and RRSE, reached more than 100 %, where the algorithm was not able to predict the valid values because the values were constant for all the input data.

TABLE II. STATISTICAL PERFORMANCE RESULTS FOR THE ML ALGORITHMS USING 10 FOLDS CROSS-VALIDATION TESTING MODE

CLASSIFIER	R^2	MAE	RMSE	RAE	RRSE
ANN	0.1584	6.9869	14.7296	115.6751	134.899
Random Forest (RF)	0.4995	4.4208	9.3905	73.1899	86.0011
Random Tree (RT)	0.1317	5.5088	12.2059	91.2029	111.7856
Decision Table (DT)	0.2663	4.9232	11.2294	81.5082	102.8431
Linear regression (LR)	0.7594	6.5889	11.1759	109.0851	102.3526
Gaussian Processes (GP)	0.6794	4.6329	7.9437	76.7025	72.7514
SMOreg	0.7383	4.3159	7.4378	73.1085	68.1179
M5P	0.6989	4.3618	7.7834	72.2135	71.2835

The distribution graph of the actual and predicted values of the SMOreg classifier are shown in Fig. 2. The graph has two dimensions, where the dataset instance's number is represented on the x-axis. On the other hand, the actual and predicted defect values are represented on the y-axis. Fig. 1 also showed that the predicted defects' values are almost the same as the actual defects' values in most 145 instances. Therefore, the error rate of the performance metrics is decreased.

Fig. 2. The distribution of the predicted and actual defect values using the best classifier, SMOreg

Scored results of the eight ML algorithms using percentage split testing mode are shown in Table III. Different percentages for the training set and the testing set ranging from 60% to 90% were implemented. MAE statistical metric has been used as an evaluation metric for the assessment performance. We can conclude that as the percentage set is increased, the error rate will be decreased. The random forest (RF) algorithm achieved the lowest MAE error rate for both 60% and 70% training sets. On the other hand, the random tree (RT) algorithm achieved the lowest MAE error rate for both 80% and 90% training sets.

Figure 7: ML algorithms appearing in Assim, et al. (2020) in [1]

discussion on the pre-processing techniques that were used. Lastly, immediately after the paper reports the ML algorithm accuracies, the paper goes straight into the conclusion. Upon reviewing further details of the paper, it seems that the model accuracies were obtained by testing the data on a public data set (the NASA promise repository). The paper does not provide any evidence of real-world impact in the original domain problem: software defect prediction. Therefore, the answer of "No" is recorded for RQ2.

Another example of applying the classification scheme is done on Kumar, et al. (2018) [3]. Again, reviewing the abstract shows keywords like "Random Forest" (a ML algorithm), as well as statistical terms like SMOTE, which indicate the paper is likely to provide accuracy metrics. Again, review of the body is justified. In this example there is no explicit methodology section the paper, but in scanning the paper, the steps of the ML pipeline are stated. The steps in the ML pipeline include data sampling, feature ranking, model selection, training, and performance evaluation. Examining the experimental results section of the paper reveals that AUC is used throughout. At this point, answers are recorded for RQ1, RQ3, and RQ4. Both of these results are recorded int he Google doc. Answering RQ2 required further examination of the paper. Upon reading the study's goals, it is clear that the study does lean towards producing some quantifiable evidence of the impact of the ML study. A goal of the paper is: "To conduct an in-depth empirical analysis on real-world web-services dataset and investigate the performance of the machine learning algorithms and object oriented source code metrics based features for predicting web- services anti-patterns. To examine the relative performance of various classification algorithms, feature ranking techniques, data sampling methods to encounter the class imbalance problem and object oriented source code metrics using Area Under an ROC Curve (AUC) and statistical hypothesis based testing approaches" [3]. The paper does conduct an empirical study, however, the context is on the evaluation of ML model accuracy on real-world data sets to assert if there is a statistical difference between the results of different ML algorithms. The answer to RQ2 in this case is "No".

This next example is one which fell immediately into exclusion through meeting one of the exclusion criteria. The paper is by Nadgowda, et al. (2017) [4]. The reason for the exclusion of this work was that the paper was purely software-based with no ML study being conducted, despite containing the keywords in the search. The last example is part of a class of papers which were all excluded because they were not software-based or software-centered. The study by Al-Tameemi, et al. (2020) is titled: "Predictive Learning Analytics in Higher Education" [5]. The reason this study made it to the search results was likely that students use "software" in higher learning. Other similar exclusions were papers that were based on finance, astrophysics, and geophysics as the domain-problem in question. These were added to the keyword search as a NOT condition as described previously.

Finally, below are the results summary of the initial 42 papers for which the classification scheme was applied. Note that these 42 papers included those which were excluded (and therefore did not make it into the final calculation). Note also that as of this midterm report submission, the original "yes/no" categories have been used. The midterm presentation will contain results according to the more detailed categorical bins that were used.

RQ1	RQ2	RQ3	RQ4
0.684	0.368	0.789	0.684

From the above, approximately 68.4% of those papers reviewed that were not excluded present some accuracy measure, while 36.8% quantify the impact and benefit of the ML study in the original domain context, 78.9% present 3 or more steps of the ML pipeline in their study, and 68.4% of studies use at least 2 ML algorithms. These results were based on [1] to [34] in the references.

REFERENCES

[1] M. Assim, Q. Obeidat and M. Hammad, "Software Defects Prediction using Machine Learning Algorithms," 2020 International Conference on Data Analytics for Business and Industry: Way Towards a Sustainable Economy (ICDABI), 2020, pp. 1-6, doi: 10.1109/ICDABI51230.2020.9325677.

[2] James Ivers, Ipek Ozkaya, Robert L. Nord, and Chris Seifried. 2020. Next generation automated software evolution refactoring at scale. In Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2020). Association for Computing Machinery, New York, NY, USA, 1521–1524. DOI:https://doi.org/10.1145/3368089.3417042

[3] L. Kumar and A. Sureka, "An Empirical Analysis on Web Service Anti-pattern Detection Using a Machine Learning Framework," 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC), 2018, pp. 2-11, doi: 10.1109/COMPSAC.2018.00010.

[4] S. Nadgowda, S. Duri, C. Isci and V. Mann, "Columbus: Filesystem Tree Introspection for Software Discovery," 2017 IEEE International Conference on Cloud Engineering (IC2E), 2017, pp. 67-74, doi: 10.1109/IC2E.2017.14.

[5] G. Al-Tameemi, J. Xue, S. Ajit, T. Kanakis and I. Hadi, "Predictive Learning Analytics in Higher Education: Factors, Methods and Challenges," 2020 International Conference on Advances in Computing and Communication Engineering (ICACCE), 2020, pp. 1-9, doi: 10.1109/ICACCE49060.2020.9154946.

[6] D. Zhang, Y. Dang, S. Han and T. Xie, "Teaching and Training for Software Analytics," 2012 IEEE 25th Conference on Software Engineering Education and Training, 2012, pp. 92-92, doi: 10.1109/CSEET.2012.14

[7] A. Londhe and P. P. Rao, "Platforms for big data analytics: Trend towards hybrid era," 2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS), 2017, pp. 3235-3238, doi: 10.1109/ICECDS.2017.8390056.

[8] K. Al-Gumaei, A. Müller, J. N. Weskamp, C. S. Longo, F. Pethig and S. Windmann, "Scalable Analytics Platform for Machine Learning in Smart Production Systems," 2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), 2019, pp. 1155-1162, doi: 10.1109/ETFA.2019.8869075.

[9] A. Aleksieva-Petrova, V. Gancheva and M. Petrov, "Software Architecture for Adaptation and Recommendation of Course Content and Activities Based on Learning Analytics," 2020 International Conference on Mathematics and Computers in Science and Engineering (MACISE), 2020, pp. 16-19, doi: 10.1109/MACISE49704.2020.00010.

[10] H. H. Huang and H. Liu, "Big data machine learning and graph analytics: Current state and future challenges," 2014 IEEE International Conference on Big Data (Big Data), 2014, pp. 16-17, doi: 10.1109/BigData.2014.7004471.

[11] S. Juddoo and C. George, "A Qualitative Assessment of Machine Learning Support for Detecting Data Completeness and Accuracy Issues to Improve Data Analytics in Big Data for the Healthcare Industry," 2020 3rd International Conference on Emerging Trends in Electrical, Electronic and Communications Engineering (ELECOM), 2020, pp. 58-66, doi: 10.1109/ELECOM49001.2020.9297009.

[12] Q. Bi et al., "Software Architecture for Machine Learning in Personal Financial Planning," 2020 Intermountain Engineering, Technology and Computing (IETC), 2020, pp. 1-4, doi: 10.1109/IETC47856.2020.9249171.

[13] B. Dayyani, "Software architecture design and development of multi-layer highly modular platform using intelligent components for dynamic big data analytics," 2016 4th International Symposium on Computational and Business Intelligence (ISCBI), 2016, pp. 45-53, doi: 10.1109/ISCBI.2016.7743257.

[14] A. Kanawaday and A. Sane, "Machine learning for predictive maintenance of industrial machines using IoT sensor data," 2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS), 2017, pp. 87-90, doi: 10.1109/ICSESS.2017.8342870.

[15] N. Bosch and J. Bosch, "Software Logs for Machine Learning in a DevOps Environment," 2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), 2020, pp. 29-33, doi: 10.1109/SEAA51224.2020.00016.

[16] R. Malhotra, L. Bahl, S. Sehgal and P. Priya, "Empirical comparison of machine learning algorithms for bug prediction in open source software," 2017 International Conference on Big Data Analytics and Computational Intelligence (ICBDAC), 2017, pp. 40-45, doi: 10.1109/ICBDACI.2017.8070806.

[17] S. Poudyal, Z. Akhtar, D. Dasgupta and K. D. Gupta, "Malware Analytics: Review of Data Mining, Machine Learning and Big Data Perspectives," 2019 IEEE Symposium Series on Computational Intelligence (SSCI), 2019, pp. 649-656, doi: 10.1109/SSCI44817.2019.9002996.

[18] B. Zhang, "A Collective Communication Layer for the Software Stack of Big Data Analytics," 2016 IEEE International Conference on Cloud Engineering Workshop (IC2EW), 2016, pp. 204-206, doi: 10.1109/IC2EW.2016.35.

[19] T. P. Pushphavathi, "An approach for software defect prediction by combined soft computing," 2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS), 2017, pp. 3003-3006, doi: 10.1109/ICECDS.2017.8390007.

[20] S. Nahar, T. Zhong, H. N. Monday, M. O. Mills, G. U. Nneji and H. S. Abubakar, "A Survey on Data Stream Mining Towards the Internet of Things Application," 2019 4th Technology Innovation Management and Engineering Science International Conference (TIMES-iCON), 2019, pp. 1-5, doi: 10.1109/TIMES-iCON47539.2019.9024597.

[21] E. Linstead, R. Burns, D. Nguyen and D. Tyler, "AMP: A platform for managing and mining data in the treatment of Autism Spectrum Disorder," 2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), 2016, pp. 2545-2549, doi: 10.1109/EMBC.2016.7591249.

[22] H. K. Dam, T. Tran and A. Ghose, "Explainable Software Analytics," 2018 IEEE/ACM 40th International Conference on Software Engineering: New Ideas and Emerging Technologies Results (ICSE-NIER), 2018, pp. 53-56.

[23] S. Pradhan, V. Nanniyur and P. K. Vissapragada, "On the Defect Prediction for Large Scale Software Systems – From Defect Density to Machine Learning," 2020 IEEE 20th International Conference on Software Quality, Reliability and Security (QRS), 2020, pp. 374-381, doi: 10.1109/QRS51102.2020.00056.

[24] M. Kim, T. Zimmermann, R. DeLine and A. Begel, "The Emerging Role of Data Scientists on Software Development Teams," 2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE), 2016, pp. 96-107, doi: 10.1145/2884781.2884783.

[25] M. Staples, L. Zhu and J. Grundy, "Continuous Validation for Data Analytics Systems," 2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C), 2016, pp. 769-772.

[26] J. F. Low, T. Yathog and D. Svetinovic, "Software analytics study of Open-Source system survivability through social contagion," 2015 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), 2015, pp. 1213-1217, doi: 10.1109/IEEM.2015.7385840.

[27] V. Mir Khatian, Q. Ali Arain, M. Alenezi, M. Owais Raza, F. Shaikh and I. Farah, "Comparative Analysis for Predicting Non-Functional Requirements using Supervised Machine Learning," 2021 1st International Conference on Artificial Intelligence and Data Analytics (CAIDA), 2021, pp. 7-12, doi: 10.1109/CAIDA51941.2021.9425236.

[28] A. Moin, "Data Analytics and Machine Learning Methods, Techniques and Tool for Model-Driven Engineering of Smart IoT Services," 2021 IEEE/ACM 43rd International Conference on Software Engineering: Companion Proceedings (ICSE-Companion), 2021, pp. 287-292, doi: 10.1109/ICSE-Companion52605.2021.00130.

[29] S. Brandys, U. Cakmak, L. Cmielowski and M. Solarski, "From model building to analytics solution in hours the enterprise platform for analytics teams," 2017 International Conference on Behavioral, Economic, Socio-cultural Computing (BESC), 2017, pp. 1-3, doi: 10.1109/BESC.2017.8256382.

[30] M. Z. Nasrabadi and S. Parsa, "Learning to Predict Software Testability," 2021 26th International Computer Conference, Computer Society of Iran (CSICC), 2021, pp. 1-5, doi: 10.1109/CSICC52343.2021.9420548.

[31] L. E. Lwakatare, E. Rånge, I. Crnkovic and J. Bosch, "On the Experiences of Adopting Automated Data Validation in an Industrial Machine Learning Project," 2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP), 2021, pp. 248-257, doi: 10.1109/ICSE-SEIP52600.2021.00034.

[32] G. R. Khazankin, S. Komarov, D. Kovalev, A. Barsegyan and A. Likhachev, "System architecture for deep packet inspection in high-speed networks," 2017 Siberian Symposium on Data Science and Engineering (SSDSE), 2017, pp. 27-32, doi: 10.1109/SSDSE.2017.8071958.

[33] S. Nadgowda, S. Duri, C. Isci and V. Mann, "Columbus: Filesystem Tree Introspection for Software Discovery," 2017 IEEE International Conference on Cloud Engineering (IC2E), 2017, pp. 67-74, doi: 10.1109/IC2E.2017.14.