Impact of Machine Learning on Software Development Life Cycle

Maryam Navaei and Nasseh Tabrizi
Department of Computer Science, East Carolina University, East 5th St., Greenville, NC, U.S.A.

Keywords: Software Engineering, Software Development Life Cycle, Artificial Intelligence, Machine Learning, Machine Learning Algorithms.

Abstract: This research concludes an overall summary of the publications so far on the applied Machine Learning (ML) techniques in different phases of Software Development Life Cycle (SDLC) that includes Requirement Analysis, Design, Implementation, Testing, and Maintenance. We have performed a systematic review of the research studies published from 2015-2023 and revealed that Software Requirements Analysis phase has the least number of papers published; in contrast, Software Testing is the phase with the greatest number of papers published.

1 INTRODUCTION

The dominant industrial sector is software engineering (SE). Therefore, automating SE is the most important task of the present. Artificial intelligence (AI) has the potential to strengthen SE in that way (Bhagyashree et al.,2015). To ensure delivery of the anticipated level of service, software systems must be able to adapt to changing circumstances (Elhabbash et al., 2019). Therefore, software systems need to autonomously adapt to changing environments to guarantee expected quality of service (Wan et al, 2019).

The demand for increased automation and intelligence has sparked new developments in ML and AI. Researchers are using machine learning (ML) and AI more frequently to solve defects and deficiencies in software systems and the software development life cycle (SDLC). (Khomh et al., 2018). ML is a subfield of AI that raises software system performance standards by using data or prior development expertise. (Zhang et al., 2017, Zhu et al., 2018).

With the use of ML, computer systems may imitate some aspects of human intelligence while using data, examples, and experience without human intervention. Computer systems may learn rules from data with the aid of ML algorithms (England, 2018). These technologies will be improved such that they serve as the main building blocks of several software-intensive systems. The Information Technology sector is very interested in incorporating AI capabilities into software and services as a result of recent developments in ML. Systems that make use of ML technology have unique traits that set them apart from other software systems. The effectiveness of the training datasets used to develop the prediction models has a significant impact on the functionalities of ML-based systems. The learning outcomes and, consequently, the functional behavior of programs is significantly impacted by changing the training dataset (Bird et al., 2017, Nakajima, 2018).

By assisting computer systems in learning from experience how to carry out a certain activity autonomously, machine learning is used to constantly enhance system performance and efficiency on a given task. The SDLC heavily utilizes ML, which has been used in other fields (Alloghani et al., 2020, Shafiq et al., 2021, Abubakar et al., 2020). ML has emerged as the preferred method for developing useful software systems in a variety of settings, including computer vision, speech recognition, natural language processing, robot control, health systems, banking, defense, e-commerce, and many more (Panichella and Ruiz, 2020, Bhatore et al., 2021).

In several areas, such as behavior extraction, testing, and problem repair, ML approaches are being utilized to speed up the SDLC (Meinke and Bennaceur, 2018, DE-Arteaga et al., 2018). As a result, technological invention has adopted a new paradigm. The development team must devote a lot of time throughout the SDLC to debating how the system must function to determine which features should be prioritized and which ones should be

Navaei, M. and Tabrizi, N.
Impact of Machine Learning on Software Development Life Cycle.
DOI: 10.5220/0011997200003464
In *Proceedings of the 18th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2023)*, pages 718-726
ISBN: 978-989-758-647-7; ISSN: 2184-4895

dropped. This requirement analysis must be followed by the phases of design, implementation, testing, and maintenance (Cetiner and Sahingoz, 2018).

Software developers may use ML algorithms to speed up the decision-making process based on prior projects to provide data-driven business decisions that eventually aid in lowering development risks and expenses. For software firms, this is priceless. It might be difficult to create an exact budget and plan for developing software systems since projects can run beyond budget and deadlines. As a result, accurate effort estimation is essential for software project planning. The development team may use ML approaches to examine data from previous projects to produce a more accurate budget estimate and delivery time frame (Yurdakurbann and Erdogan, 2018).

Developers may automatically maintain the code and evaluate it using ML approaches. This leads to the creation of Software Systems with a high level of quality since maintenance is one of the most important but costly parts of the SDLC. Rapid prototyping, intelligent programming assistants, automatic analytics and error handling, automatic code rewriting, exact estimations, and strategic decision-making might indeed benefit from ML models in traditional SDLC (Khomh et al., 2018, Abubakar et al., 2020). Even though ML datasets for SDLC research are frequently poorly managed, documented, and lack defined generation methods (Hutchinson et al., 2021) The use of ML methods has several benefits.

Due to the importance of ML approaches in the SDLC and their benefits, we have produced a comprehensive analysis to examine how data analytics and ML algorithms affect each stage of the SDLC. We were looking for phases in which ML models were commonly used. Our objective was to determine why ML techniques in some phases have received more attention than others. We also sought to see whether ML methods may be used in less well-known stages.

We gathered and updated our previous database of over 300 journal articles and conference papers from the last seven years that were obtained from the ACM, IEEE, and Springer digital libraries to prepare our review. In order to illustrate graphically and summarize current research trends, we generated figures. Our study reveals notable variations between the SLDC and software engineering's application of ML methods (e.g., Requirements Analysis, Architecture Design, Implementation, Testing and Maintenance). The findings for each phase of the SDLC are discussed after a review of similar work and our approach. We then wrap up by summarizing our findings and making recommendations for more study.

2 RELATED WORK

This article extents our previous reviews on recent studies on the use of ML in various SDLC phases and SE broadly. Several earlier academics have carried out practical investigations of SE for data science due to the prominence of integrating ML methods into SDLC. In 2022, We studied 300 publications with the focus on eight most algorithms used in SDLC (Navaei and Tabrizi, 2022). One research compared how ML systems and non-ML systems developed in diverse SE domains. As none of the SE stages have an established set of tools and methodologies, that study employed interviews to identify pain points from a general tooling position to learn about the difficulties and obstacles of deploying visual analytic tools (Kim et al., 2018, Giray, 2021). Another research characterized the roles and practices of professionals in data science (Wan et al, 2019).

Even when the learning technique is used correctly, faulty training data might occasionally cause the ML model to be faulty. For these systems, conventional testing methods are insufficient. Software engineers must thus do research and provide cutting-edge methods of reporting these difficulties (Khomh et al., 2018). Software engineers concentrated on the difficulty of testing and ensuring the correctness of systems created using ML and AI models.

Applying ML algorithms to each stage of the SDLC has been the subject of several recent literature studies (Talele and Phalnikar, 2021, Lima et al.,2020, Sobhy et al., 2021, Syaeful et al.,2017). Although their review was conducted for the period of 1991 to 2021, we found a paper that gives a larger context and examines the link of ML approaches and its tools within SDLC phases, which is somewhat similar to our work (Shafiq et al., 2021). To analyze the influence of ML throughout the whole SDLC in the most recent papers, our work, however, employs a systematic review technique (2015-2023) compared to the majority of publications' single-phase emphasis. The discovery of the eight most common ML algorithms throughout the whole SDLC is another contribution made by our study. To determine their total popularity based on each stage of their life cycle, we analyzed those algorithms individually in each phase of the cycle.

3 METHODOLOGY

Based on interests shown in our recent publication, we decided to expand our research to concentrate on works published in (2015-2022). We ultimately made the decision to update our evaluation on the journal articles and conference papers published during the past seven years. According to the number of publications and refer to our latest research, we discovered that the use of ML in SDLC has significantly increased year by year even though it had a slow start in the first half of last decade. This is due recent developments in ML and their significant impact on SDLS since ML techniques can be used to modify and update software systems.

We searched the same databses: ACM, IEEE, and Springer using various keywords and queries to find our findings. Some inquiries returned the results we were looking for in IEEE and Springer, but we had to change our keywords and queries to get the results we wanted from ACM. Additionally, we discovered that utilizing verbatim searches like those used in the other two libraries would cause ACM to return many articles that were unrelated. As a result, when searching the ACM database, we had to add extra restrictions. To guarantee that our query returned results that were pertinent, for example, we had to add the phrase “Artificial Intelligence”.

In several cases, our inquiries brought up the same or improved work in many publications. Because each publication focused on a different subject, we classified each of those cases as a new publication. Additionally, there were instances in which one ML approach was used in numerous phases; we counted those individually for each phase. We extend our research to study the eight of the most popular ML algorithms in each phase to get a more concentrated observation as a result of the latter situation.

4 CURRENT APPLICATIONS OF MACHINE LEARNING TO SDLC

ML plays a significant role throughout the SDLC. In this section we discuss the impact of applying ML in each phase of the SDLC:

- Software Requirements Analysis
- Software Architecture Design
- Software Implementation
- Software Testing
- Software Maintenance

We provide an overview of each phase of the SDLC and discuss current research trends in ML for each phase.

4.1 Software Requirements Analysis and Machine Learning

Clarifying the specifications required for a software system to satisfy the business requirements is the goal of software requirements analysis, also known as requirements engineering. Clear software requirements (SR) that describe the needs and expectations of the product in great details are a crucial component of high-quality software development. (Navarro-Almanza et al., 2017).

There are two primary phases of requirements engineering (RE), which is a crucial part of the whole software development process. Prioritization and Identification of Requirements (Talele and Phalnikar, 2021). To what extent a software system succeeds or fails depends on requirements analysis. Requirements classification is another task that software engineers must complete during the analysis stage. Software Engineers were able to automate the process of classifying the requirements into functional requirements after employing ML methods (FRs) and Non-Functional Requirements (NFR) (Quba et al., 2021). RE is also one of the key factors in Software Quality (Panichella and Ruiz, 2020).

Requirement analysis in the SDLC refers to the processes of accurate requirement elicitation, effective requirement inspection, and transparent requirement documentation. Requirement Engineering becomes a hard problem that drives up development costs as software systems get more complicated and large-scale. Engineers' interest in automated requirement analysis approaches has increased as a result, and these techniques may help to analyze SR more quickly and accurately, which might save development costs. Engineers must choose which set of needs to evaluate first when doing requirement prioritization, and this decision is made using machine learning approaches. The classification and prioritization of the needs using ML algorithms is likely to be effective, and so is the method of evaluation (Talele and Phalnikar, 2021).

Researchers have shown a lot of interest in applying Supervised ML algorithms including Support Vector Machine, Naïve Bayes, Decision Tree, K-Nearest Neighbor and Random Forest (Gramajo et al., 2020).

RE plays a key role in the success of a project. Based on previous research that was conducted on 350 companies to understand project failure rates, 16.2%

projects were completed successfully, 52.7% faced challenges and were partially completed. About 31% of the projects were never completed due to poor RE (Haleem et al., 2021).

There aren't many public ML datasets accessible for requirements analysis, which is one of their key drawbacks. The PROMISE repository, one of the most popular RE databases, is imbalanced and has only 625 categorized requirements stated in plain language (Iqbal et al., 2018). Despite limitations of current ML algorithms in recognizing and prioritizing requirements, including scalability, dependency, and complexity (Talele and Phalnikar, 2021), The crucial function that these methods serve in assisting developers to document their software more accurately is what motivates academics to continue showing interest in using ML algorithms in SR categorization. As a result, the software system is simpler to use and comprehend (Quba et al., 2021).

4.2 Software Architecture Design and Machine Learning

The established requirements from the Analysis phase are transformed into an implementable form during the Architecture Design phase, which is sometimes referred to as the preliminary or general design stage. Software quality is significantly influenced by software design. Anti-patterns, high dependence design, and enormous source code files are a few examples of poor design. These complicate SE responsibilities by increasing the difficulties and costs of addressing software flaws (Zhang et al., 2017, McIntosh and Kamei 2017).

To satisfy all system requirements, software design must be transparent and easy to comprehend. Many businesses and scholars are interested in figuring out how the link between Software Design and Maintenance works since later phases of the SDLC heavily rely on the Architecture Design phase. Researchers have been attempting to introduce several ML approaches for effective prediction of the influence of software design on system maintainability (Elmidaoui et al., 2020). In this stage, a variety of ML approaches are being used to problems like anticipating the user interface (UI) design of a mobile application, anticipating the architecture of safety-critical systems, detecting anti-patterns in web services, etc. Design patterns, for instance, might increase complexity. As a result, maintenance and evolution activities rise. By utilizing ML for design pattern recognition, the complexity of the system is reduced, and the architecture and design of the program are made clearer (Mhawish and Gupta, 2019, Dwivedi et al., 2016, Dwivedi et al., 2016).

Since detecting Code Anomalies requires refactoring approach, Support Vector Machine is the most effective ML algorithm used for this matter (Gramajo et al., 2020). Data mining from design patterns, which is based on supervised learning and software metrics, is another of the most well-liked study areas in the design phase. By eliminating as many erroneous matches as feasible, ML approaches are utilized to improve pattern mining. This approach mostly utilizes ML algorithms as Layer Recurrent Neural Network, Random Forest, Support Vector Machine, and Boost (Navarro-Almanza et al., 2017, Giray, 2021).

4.3 Software Implementation and Machine Learning

The Implementation phase involves construction of the actual software system. Coding the system takes place in this phase and when it is complete, the result will be evaluated against the elicited requirements from the Analysis phase. Software defects are prevalent in software development and might cause several problems for users and developers alike. As a result, research has examined distinct techniques to diminish the impacts of these defects in the source code. One of the most prominent algorithms focuses on defect prediction using ML methods. This helps developers in managing these defects before they are commenced in the production (Esteves et al., 2020).

One of the prominent study topics for this stage is code refactoring. Refactoring modifies the system in a way that doesn't affect the code's external behavior but enhances its internal organization. Software with code smells is a hint that the SDLC's design and coding standards have not been followed. If code smells are not addressed, this problem might grow since it will take exponentially more work to fix the problems (Aniche et al., 2020).

The best method for forecasting code refactoring is known to be supervised machine learning techniques, which can aid developers in making quicker and more informed judgments on what to alter. Support Vector Machines, Naive Bayes, Decision Trees, Random Forest, Logistic Regression, and Neural Networks were among the ML methods studied. Using Apache, F-Droid, and GitHub datasets, Random Forest has emerged as the best accurate method for forecasting software refactoring (Sagar et al., 2021).

4.4 Software Testing and Machine Learning

Engineers must identify a system's flaws and problems to evaluate and validate software systems. Thus, one of the most crucial stages in the SDLC is software testing. Software testing's goal is to identify flaws, especially during automated testing (Nasrabadi and Parsa, 2021). There should be automated tests to find these mistakes prior to a system being deployed since errors, failures, and defects might occur due to frequent modifications to a system's codebase. Finding system flaws might result in improvements that save expenses by up to a third (Lima et al., 2020).

Developers use testing to make sure a system behaves as intended. To ensure time effectiveness, effort, and money consumption, testing activities must be anticipated (Syaeful et al., 2017).

As software systems increase in complexity, it becomes more difficult to detect faults (Li et al., 2020). During the software testing process, we may gather a variety of data types, such as execution traces, coverage details for test cases, failure data, etc. When used to software testing, machine learning algorithms find patterns in data that frequently relate to the data generating process and are then utilized for decision-making. To convert unstructured test data into a set of abstract test scenarios, experts must contribute (Bhavsar et al., 2019).

To give test output information, test cases are created. In Software Testing, decision trees or induction rules are frequently utilized, particularly for fault prediction. Compared to more advanced approaches like neural networks or support vector machines, models created by these techniques are easier to understand. For these datasets (the most popular Testing dataset is from NASA), Support Vector Machine and Random Forest offer the best prediction models for fault prediction (Bhavsar et al., 2019, Zhu, 2020).

Techniques that suggest unique interactions inside a system have a lot to offer when it comes to software testing. However, methods like random testing, fuzzing, and exploratory testing have the drawback of requiring software engineers to manually review the outputs of the tests to ensure they are right, which adds to their workload (Roper, 2019). For assessing software usability at the moment, testing methods and software reengineering are necessary and most important. (Banga et al., 2019, Herzig and Nagappan, 2015).

Due to the quantifiable nature of Software Testing, there are many datasets available. Likewise, the number of publications is also significantly higher compared to other phases.

4.5 Software Maintenance and Machine Learning

Software maintenance is the practice of continuing to alter and update a software system after it has been deployed to fix bugs. The SDLC's last phase is this one. In SE, software maintenance has grown to be a crucial component of software quality. Therefore, accurate and timely prediction of this feature is a vital prerequisite for effective management throughout the SDLC's final phase (Gupta and Chug, 2021).

Software maintenance includes a system's ongoing optimization as well as repair and preservation. Four major categories make up this phase: preventative maintenance, corrective maintenance, adaptive maintenance, and perfective maintenance (Sulaiman, 2005). Software Maintainability Prediction has been investigated using a broad range of ML approaches, including hybrid, ensemble, and meta-heuristic techniques (SMP) (Alsolai, 2018, Gupta and Chug, 2021, Baskar and Chandrasekar, 2018). Yet researchers continue to seek improvement in this area.

About 60 to 70 percent of the overall cost of the SDLC is occasionally devoted to maintenance (Gupta and Chug, 2021). Maintenance has grown more challenging as software systems have grown in complexity and scale. Because of this, maintaining software has become a significant problem for businesses. Software upkeep is closely related to budgetary implementation and project success (Jha et al., 2019).

The most often employed ML method in software maintenance prediction models is fuzzy logic (Haleem et al.,2021). Other techniques include ML algorithms such as Soft Computing, Fuzzy Networks, Evolutionary Algorithm, Deep Learning and ID3 (Xin et al., 2018).

5 RESULTS

We include publications from 2015 to 2023 in this review.

The figure below summarizes how many journal articles and conference papers are published in ACM, IEEE and Springer for the period of 2015 to 2023 that focus on ML techniques applied in each phase of SDLC.

Table 1 lists the keywords we used to retrieve these publications and ultimately generate the figures to show the overall results. For some of the phases, we had to modify our search queries as it was a quite challenge to find more related and accurate results. Whereas some phases like Testing, had a very large number of publications.

Table 1: Searched Keywords for Paper Retrieval.

Keywords (Searched Queries)
Software engineering AND Machine learning
Software Development Life Cycle AND Machine Learning
Machine Learning AND Software System Requirements Analysis
Machine Learning AND Software in Each Phase
Impact of Machine Learning in Each Phase
Software Requirements Analysis AND Each Algorithm
Software Design Phase AND Each Algorithm
Software Testing AND Each Algorithm
Software Maintenance AND Each Algorithm

As shown in Figure 1, Software Requirements Analysis has the least number of papers published in this topic; in contrast, Software Testing has the largest number of publications. The reason is partially because there are many datasets available for Software Testing due to its quantifiable nature as well as the ease of obtaining data in this phase.

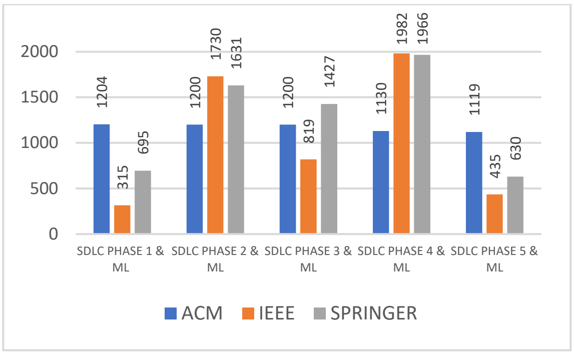

Figure 1: ML Applied in SDLC – Papers published from 2015 to 2023.

Hence it is much easier to utilize ML techniques in Software Testing than in Requirements Analysis.

In Figure 2, we narrowed our findings by selecting eight ML algorithms in all phases of SDLC (Random Forest, Support Vector Machine, Naïve Bayes and Decision Trees, Logistic Regression, Reinforcement Learning, K Nearest Neighbor and Gradient Boosting). Ultimately, we analyzed all the relevant publications done in ACM, IEEE and Springer in the same timeframe to see which ML methodologies are the most popular in particular phases.

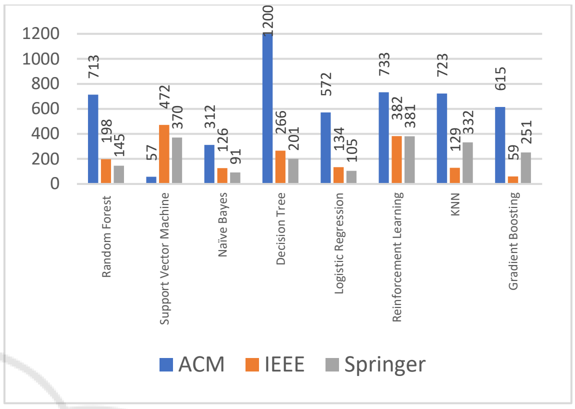

Figure 2: SDLC Software Requirements Analysis - ML Applied in SDLC – Papers published from 2015 to 2023.

Based on the results shown in Figure 2, Decision Trees are most popular algorithms among the other elected techniques. One reason is because Decision Trees are quick to create and help with eliminating dead ends. This decreases the probability of errors made in this phase that can affect subsequent phases as well. This technique can greatly simplify the SR gathering process and saves engineers significant time and budget resources. Another popular ML techniques is Reinforcement Learning which is used on understanding the relationships among software requirements by generating traceable links between high-level and low-level requirements (Shafiq et al.,2021, Rezaei and Tabrizi, 2021).

In the process of SR, due to their exclusive characteristics, the appropriate data sets are extremely dimensional, sparse, and are mostly the outcome of ambiguous expressions and, consequently show problematic challenges for data processing techniques (Borges et al., 2021). there are many repositories with duplicate code, which constitutes data inconsistency (Allamanis, 2018). Hence, more work needs to be done on creating more diverse and appropriate datasets for this phase.

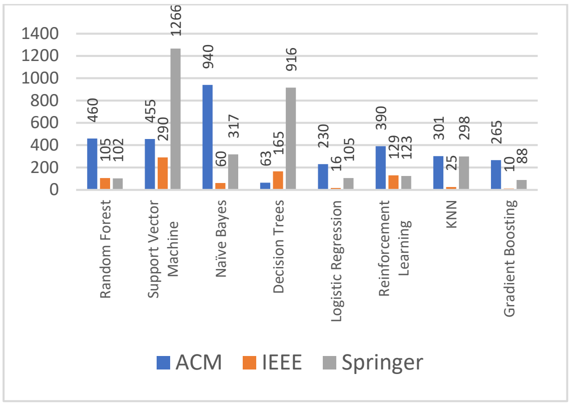

Figure 3: SDLC Software Architecture Design - ML Applied in SDLC – Papers published from 2015 to 2023.

Support Vector Machine (SVM) is one of the most popular ML techniques used in SDLC. It is a supervised learning algorithm used for both classification and regression problems. This technique is very accurate and robust in detecting design anomalies such as code smells also known as Bad Smells, Design Flaws, Anti Patterns and Code Anomaly (Zhang et al., 2006).

ML demands massive datasets to train on and these should be unbiased and have great quality which sometimes requires new data to be generated. We suggest applying ML algorithms more in Software architecture design as little work has been done and needs to be explored more (Borges et al., 2021).

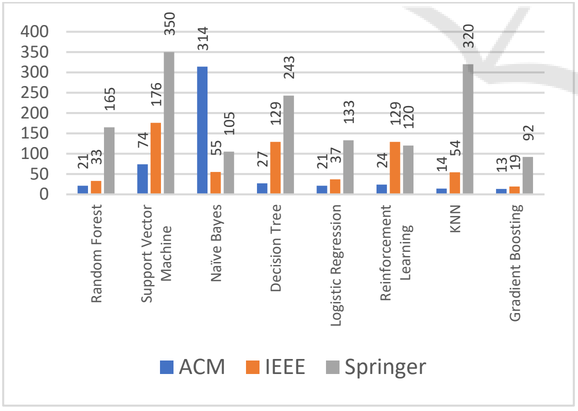

Figure 4: SDLC Phase Implementation - ML Applied in SDLC – Papers published from 2015 to 2023.

Again, Support Vector Machine is most frequent algorithm used in the third phase of Software Development Life Cycle known as Software Implementation. This technique is very popular in Design and Development of software applications.

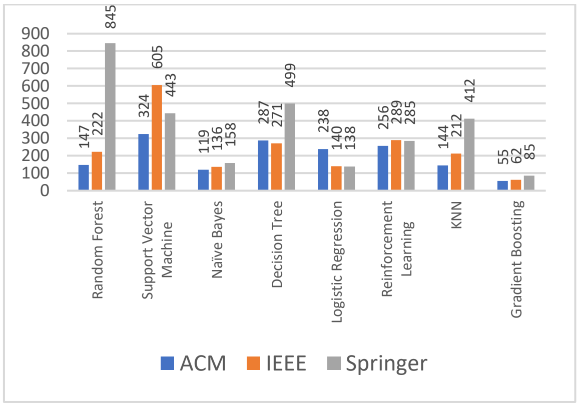

Figure 5: SDLC Phase Testing- ML Applied in SDLC – Papers published from 2015 to 2023.

As shown in Figure 5, both Random Forest and Support Vector Machine are popular techniques in Software Testing phase. However, SVM has slightly more publications. One reason for this is SVM is the better classifier to eliminate infeasible test cases saving time and costs in projects which is a huge achievement as testers spend more time and their resources on testing mobile and hybrid applications. ML helps testers to better understand user's needs and respond faster to the everchanging expectations by improving automation testing, reduced UI based testing, assisting in API testing, and improving accuracy. Right after those two algorithms, Decision Trees and Reinforcement Learning play a significant role in this phase.

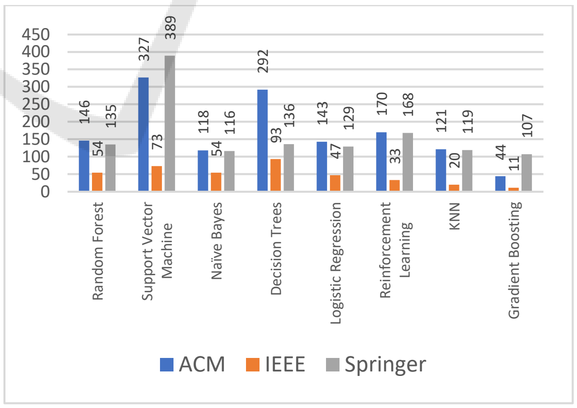

Figure 6: SDLC Maintenance - ML Applied in SDLC – Papers published from 2015 to 2023.

Data Scientists and Software Engineers have shown the most interest in applying Support Vector Machine in last phase of SDLC than other techniques. However, Decision Trees have high usage as well. Classification and refactoring approaches play a significant role when maintaining software systems.

Available datasets are not sufficient to give conclusive results which provide information regarding the input data of the system (Jha et al., 2019) and more work need to be done to create new datasets to determine the accuracy of the precision and decreasing the complexity.

6 CONCLUSIONS

Machine Learning is becoming one of the most important technologies used in Software Development Life Cycle. ML approaches are being used for inadequately implied problem domains where little knowledge exists for humans to develop effective algorithms. ML has different types: Supervised Learning, Semi-Supervised Learning, Unsupervised Learning and Reinforcement Learning. Our study shows Support Vector Machine (SVM) is one of the most popular supervised ML algorithms in current SE research. It is being applied on all the phases of SDLC as it can be used for both classification and regression problems. While SVM is not considered the best performing algorithm in all cases, it remains among the most highly used ML methods. Reinforcement Learning is gaining a lot of interests in recent works and has a high impact in requirements analysis. It has been used in Software requirements analysis for understanding the relationships between requirements at different levels of abstractions, it is an interesting topic that we can study on more in terms of simplifying SE. Software Testing is utilizing ML techniques more than any other SDLC phase since there are a lot of datasets available to researchers. Gradient Boosting is drawing lots of attention and proves to be one the most powerful algorithms (Gupta et al., 2020).

Our future work aims at addressing the challenges outlined in Software Requirements Analysis, Architecture Patterns, Software Maintenance and ultimately creation of datasets for those phases. Also, future works could look at data mining, predictive design and modeling for different software applications including mobile applications.

REFERENCES

Abubakar, H., Obaidat, M. S., Gupta, A., Bhattacharya, P., Tanwar, S. (2020) - *Interplay of Machine Learning and Software Engineering for Quality Estimations – IEEE*

Allamanis, M. (2018) - *The adverse effects of code duplication in machine learning models of code – Research Gate.*

Alloghani, M., Al-Jumeily, D., Baker, T., Hussain, A., Mustafina, J., Ahmed J. Aljaaf (2020) - *An Intelligent Journey to Machine Learning Applications in Component-Based Software Engineering* - Springer.

Software Maintainability Metrics Prediction – IEEE.

Aniche, M., Maziero, E., Durelli, R., Durelli, V. H. S. (2020) - *The Effectiveness of Supervised Machine Learning Algorithms in Predicting Software Refactoring – IEEE.*

Alsolai, H. (2018) - *Predicting Software Maintainability in Object-Oriented Systems Using Ensemble Techniques – IEEE.*

Banga, M., Bansal, A., Singh, A. (2019) - *Implementation of Machine Learning Techniques in Software*

Reliability: A framework – IEEE.

Baskar, N. and Chandrasekar, C. (2018) - *An Evolving Neuro-PSO-based Software Maintainability Prediction – IEEE.*

Bhagyashree W. Sorte, Pooja P. Joshi, Prof. Vandana Jagtap - *Use of Artificial Intelligence in Software Development Life Cycle: A state of the Art Review – Research Gate.*

Bhatore, S., Reddy, Y., R., Sanagavarapu, L. M, Chandra, S. S. (2021) - *Software Patterns to Identify Credit Risk Patterns – IEEE.*

Bhavsar, K., Gopalan, S., Shah, V. (2019) - *Machine Learning: A Software Process Reengineering in Software Development Organization – Research Gate.*

Bird, C., DeLine, R., Gall, H., Kamar, E., Nagappan, N., Nushi B., Zimmermann, T. (2017) - *Software Engineering for Machine-Learning: A Case Study – IEEE.*

Borges, O. T., Couto, J. C., Ruiz, D., Prikladnicki, R. (2021) – *Challeneges in using Machine Learning to Support Software Engineering – Research Gate.*

Cetiner, M., Sahingoz, O. K. (2020) - *A Comparative Analysis for Machine Learning based Software Defect Prediction Systems – IEEE.*

perception in autonomous driving - IEEE.

Dwivedi, A. K., Tirkey, A. , Rath, S. K. (2016) - *Applying software metrics for the mining of design pattern – IEEE.*

Dwivedi, A. K., Tirkey, A., Ray, R. B., Rath, S. K. (2016) - *Software design pattern recognition using machine learning techniques – IEEE.*

DE-Arteaga, M., Herlands, W., Neill, D. B., Dubrawski, A. (2018) - *Machine Learning for the Developing World – ACM.*

Elhabbash, A., Salama, M., Bahsoon, R., Tino, P. (2019) - *Self-awareness in Software Engineering: A Systematic Literature Review – ACM*

Elmidaoui, S., Cheikhi, L., Idri, A., Abran, A. (2020) - *Machine Learning Techniques for Software Maintainability Prediction: Accuracy Analysis – Springer.*

England, M. (2018) - *Machine Learning for Mathematical Software – Springer.*

Esteves, G., Figueiredo, E., Veloso, A., Viggiato, M., Nivio (2020) - *Understanding Machine Learning Software Defect Predictions – Springer.*

Giray, G (2021) - *A software engineering perspective on engineering machine learning systems: State of the art and challenges - Science Direct.*

Gramajo, M., Ballejos, L., Ale, M. (2020) - *Seizing Requirements Engineering Issues through Supervised Learning Techniques – IEEE.*

Gupta A., Sharma Sh., Goyal, Sh., Rashid M. (2020) – *Novel SGBoost Tuned Machine Learning Model for Software Bug Prediction - IEEE*

Gupta, S., Chug, A. (2021) - *An Optimized Extreme Learning Machine Algorithm for Improving Software Maintainability Prediction – IEEE.*

Gupta, H., Kumar, L., Neti, L. B. M. (2019) - *An Empirical Framework for Code Smell Prediction using Extreme Learning Machine – IEEE.*

Haleem, M., Farooqui, M. F., Faisal, M. (2021) - *Cognitive impact validation of requirement uncertainty in software project development – Science Direct.*

Herzig, K. and Nagappan, N. (2015) - *empirically detecting false test alarms using association rules – IEEE.*

Hutchinson, B., Smart, A., Hanna, A., Denton, E. (2021) - *Towards Accountability for Machine Learning Datasets: Practices from Software Engineering and Infrastructure – ACM.*

Iqbal, T., Elahidoost, P., Lúcio, L. (2018) - *A Bird's Eye View on Requirements Engineering and Machine Learning – IEEE.*

Jha, S., Kumar, R., Son, L. H., Abdel-Basset, M., Priyadarshini, I., Sharma, R., Long, H. V. (2019) - *Deep Learning Approach for Software Maintainability Metrics Prediction*

Karim, M. S., Warnars, H. L. H. S., Gaol, F. L., Abdurachman, E., Soewito, B. (2017) - *Software metrics for fault prediction using machine learning approaches: A literature review with PROMISE repository dataset – IEEE.*

Khomh, F., Adams, B., Cheng, J. , Fokaefs, M., Antoniol, G. (2018) - *Software Engineering for MachineLearning Applications: The Road Ahead – IEEE.*

Kim, M., Zimmermann, T., DeLine, R. and Begel, A. (2018) - *Data scientists in software teams: State of the art and challenges – IEEE.*

Li, J. J., Ulrich, A., Bai, X., Bertolino, A. (2020) - *Advances in test automation for software with special focus on artificial intelligence and machine learning – Springer.*

Lima, R. , Da Cruz, A. M. R., Ribeiro, J. (2020) - *Artificial Intelligence Applied to Software Testing: A Literature Review – IEEE*

Meinke, K., Bennaceur, A. (2018) - *Machine Learning for Software Engineering: Models, Methods, and Applications – IEEE.*

Mhawish, M. Y., Gupta, M. (2019) - *Software Metrics and tree-based machine learning algorithms for distinguishing and detecting similar structure design patterns – Springer.*

Nakajima, S. (2018) - *Quality Assurance of Machine Learning Software – IEEE.*

Nasrabadi, M. Z., Parsa, S. (2021) – *Learning to Predict Software Testability - IEEE.*

Navaei, M., Tabrizi, N. (2022) - *Machine Learning in Software Development Life Cycle: A Comprehensive Review – Research Gate*

Navarro-Almanza, R., Juarez-Ramirez, R., Licea, G. (2017) - *Towards Supporting Software Engineering Using Deep Learning: A Case of Software Requirements Classification – IEEE.*

Quba, G., Qaisi, H. A., Althunibat, A., AlZu'bi, S. (2021) - *Software Requirements Classification using Machine Learning algorithm's – IEEE.*

Panichella, S., Ruiz, M. (2020) - *Requirements-Collector: Automating Requirements Specification from Elicitation Sessions and User Feedback – IEEE.*

Rezaei, M., Tabrizi, N. (2022) – *Recommender System using Reinforcement Learning: A Survey – DBLP.*

Roper, M. (2019) - *Using Machine Learning to Classify Test Outcomes – IEEE.*

Sagar, P. S., AlOmar, E. A., Mkaouer, M. W., Ouni, A., Christian Newman (2021) - *Comparing Commit Messages and Source Code Metrics for the Prediction Refactoring Activities – Research Gate.*

Shafiq, S., Mashkoor, A., Mayr-Dorn, C., Egyed, A. (2021) - *A Literature Review of Using Machine Learning in Software Development Life Cycle Stages – IEEE.*

Sobhy, D., Bahsoon, R., Minku, L., Kazman, R. (2021) - *Evaluation of Software Architectures under Uncertainty: A Systematic Literature Review – ACM.*

Sulaiman, S. (2005) - *Viewing Software Artifacts for Different Software Maintenance Categories Using Graph Representations – Research Gate.*

Talele, P., Phalnikar, R. (2021) - *Classification and Prioritisation of Software Requirements using Machine Learning – A Systematic Review – IEEE.*

Wan, Z., Xia, X., Lo, D., Murphy, G. C. (2019) - *How does Machine Learning Change Software Development Practices? – IEEE.*

Xin, Y., Kong, L., Liu, Z., Chen, Y., Li, Y., Zhu, H., Gao, H., Hou, H., Wang, C. (2018) - *Machine Learning and Deep Learning Methods for Cybersecurity – IEEE.*

Yurdakurbann, V., Erdogan, N. (2018) - Comparison of machine learning methods for software project effort estimation – IEEE.

Zhu, H. (2018) - *Software Testing as a Problem of Machine Learning: Towards a Foundation on Computational Learning Theory – IEEE.*

Zhang, L., Tan, J., et al, D. H.. (2017) - *From machine learning to deep learning: progress in machine intelligence for rational drug discovery – IEEE.*

Zhang, X., Gu, C., Lin, J (2006) - *Support Vector Machines for Anomaly Detection – Research Gate.*

Zhang, X., Zhou, T., Zhu, C. (2017) - *An Empirical Study of the Impact of Bad Designs on Defect Proneness – IEEE.*

Zhu, Y, Chen, L, Zhou, H, Feng, W., Zhu, Q. (2018) - *Design and Implementation of WeChat Robot Based on Machine Learning – IEEE.*