MACHINE LEARNING METHODS IN SOFTWARE ENGINEERING – REVIEW

Vikas
Computer Science & Engineering, Ajay Kumar Garg Engineering College, Ghaziabad, UP
vikas@akgec.ac.in

Abstract— **This article discusses the use of machine learning (ML) in software engineering (SE) at various phases. The SDLC organises software development. Goal is to examine Examine software engineering machine learning applications. Study classifies machine learning I examined machine learning methods for software engineering projects. Evaluation, layout, execution, evaluation, and maintenance are every aspect that collectively make up the software development life cycle. Supervised and unsupervised machine learning are used to determine software requirements, design patterns, and code implementation, generation, and testing.**

Keywords—Machine Learning, SDLC, Testing, component,

I. INTRODUCTION

Hot perspectives encompass artificial intelligence (AI), machine learning (ML), data mining, and big data analytics. These fields are crucial to scientific communication. Symbolic of their influence on modern culture. Machine learning develops programmes that improve with experience. Machine learning algorithms work well. It is useful in many fields. The machine learning field has advanced. Used for feature extraction and testing in software engineering. Software developers could better understand machine learning methodologies and help users choose and execute the best methods by evaluating assumptions and assurances. Future software engineering (SE) methods and technologies will require more automation to adapt to changing software development methodologies. The system is lightweight, adaptable, and scalable to meet developers' rising needs and boost productivity. Text from the user

Increasing reliance on machine learning (ML) applications demands advanced engineering techniques to create a strong, resilient system that can adapt to future requirements. The growing dependence stresses the need for advanced and organised engineering procedures.

Software is essential to most systems and has become part of daily life. With open-source technology, networked devices, and automated processes, software systems are getting increasingly complex [1]. Software projects also involve people with different skills, which makes them more complicated.

Software faults are common since people write it. Thus, errors are unavoidable while using commercial software, especially as complexity increases [2]. These inaccuracies increase with number [3]. Automation of the Software Development Life Cycle (SDLC) utilising machine learning can help solve these problems.

II. BACKGROUND AND RELATED WORKS

The link between software engineering (SE) and machine learning (ML) has long been studied [1-3]. Some studies highlight the discrepancy between Software Engineering (SE) and ML (Machine Learning) communities, mainly due to their different specialisations. The ML community focuses on algorithm efficacy. The SE community develops and implements software-intensive systems [4]. However, cooperation between these societies benefits both. "Experts in software engineering do ML system development duties. Include designing, creating, and managing Machine Learning-assistance software. Researchers in this field seek to identify design differences between machine learning systems and traditional software to provide new methods and resources to address these disparities. However, "ML for SE" modifies AI technology to solve software engineering problems. Software defect prediction, code smell detection, reusability measurement, forecasting, and expense estimation. Researchers use machine learning models to improve software engineering utilising source code, requirement specifications, and test cases. Software engineering is about creating, using, and maintaining software development concepts. This field employs structured methods to analyse, create, execute, and manage information systems. Software development requires effective planning and organisation to ensure timely delivery and high-quality software.

Software development requires the SDLC. "Software development" encompasses the full software creation process. "SDLC" stands for Software Development Life Cycle, which includes Agile, Waterfall, DevOps, V-Model, and Iterative. Software developers use Dynamic System Development Model, Extreme Programming, and Feature Driven. Joint Application Development, Spiral, and Rapid Application Development are common software development methods. Requirements analysis and design are common SDLC processes. Execution, verification, maintenance. Stages together determine course. Software development ensures software programme implementation and functionality [5]. Software engineering has moved from waterfall to agile. The waterfall model is linear and sequential, with each phase depending on the previous ones.

III. MACHINE LEARNING IN SOFTWARE ENGINEERING

Analysis of machine learning implementation is the main goal. Developing and evaluating machine learning (ML) skills during the software development life cycle (SDLC). We have developed guiding research queries to achieve this purpose. Throughout the inquiry:
Research Question 1 (RQ1): What software programme categories exist? Does modern software development acknowledge or document it?
RQ2: Which machine learning methods were used in this software development phase?
RQ3: How are machine learning-based methods evaluated?

The entire software development process relies on requirement engineering (RE). Prioritisation and requirement identification are key to RE [6].
RQ1: Machine learning techniques define functional and non-functional software requirements [7-11].
Research Question 2: Text analysis can be done using many machine learning methods. Processing algorithms fall into two categories: Supervised learning Machine learning algorithms are divided into supervised and unsupervised categories. Semi-supervised learning (SSL) combines supervised and unsupervised classification.

A. Software design

This is the most innovative era of software development. It involves strategic planning and problem-solving for a circumstance or activity. Software developers and designers outline a repair strategy. The Software Design Document (SDD) is created at this stage. Software design is complicated. However, software design patterns improve this phase's organisation. Standardised software design patterns solve software architecture problems. ML-based approaches can discover adapter and strategy design patterns [13]. A number of studies investigated five different design patterns, including the Singleton, Adapter, Composite, Decorator, and Factory Method configurations. [14]. Agile SDLC divides the system's architecture into components. Thus, software component selection is essential to design. Multiple research reveal a novel machine learning approach [15] that can help categorise reusable software components. Experimental machine learning models used in the study are below. The chosen study uses Naive Bayes, logistic regression, random forest, neural networks, decision trees, support vector machines with various kernel functions, and decision trees.

B. Software construction

The findings show that machine learning models generate code [16, 17]. Modifying code and creating documentation [18, 19]. The selected papers demonstrate that many machine learning methods can be used for code generation. Most selected study uses Recurrent and Convolutional Neural Networks [16, 20]. Advanced conversational AI system ChatGPT uses guidance and reinforcement learning. ML models make mistakes on complex and unforeseen problems.

C. Software testing

According to ISO/IEC 24765, 2006, testing involves running a system under certain settings, documenting the results, and assessing a specific part of the system [21]. Software development requires testing. Every computer programme The product must go through several steps before being used. Experimentation helps us spot emerging difficulties. Testing allows developers to evaluate quality requirements, discover difficulties, and find proactive solutions. Machine learning (ML) in numerous software programmes for experimentation is gaining popularity [3]. The statistical software testing process included the utilisation of machine learning, specifically for the purposes of performance evaluation, testing, and the production of test cases. The model-free reinforcement learning technique known as Q-learning was utilised in testing environments that were quite complicated. Software testing is automated using Model-Inference-Driven testing (MINTest).

D. Software maintenance

Software maintenance involves improving, modifying, and correcting software for security, functionality, and performance. IEEE STD 1219-15193 defines software maintenance as changing a software product after delivery to fix bugs and improve functionality. The Systems Development Life Cycle (SDLC) concludes with product performance improvements and environmental adaptation. This stage of the SDLC distributes software to end users, who must maintain and operate it according to industry standards. At this point, machine learning is used to find software flaws. There are more capabilities that are included in maintenance software. These features include refactoring, which is the process of replacing obsolete components or algorithms with more advanced ones, updating data naming standards, and enhancing the readability of code. Several articles discuss reworking model development, including: The study "A machine learning approach to software model refactoring" offered an AI-driven method to early analyse and improve object-oriented software quality. The inquiry shows many machine learning methods at this level. Deep learning using Convolutional Neural Networks (CNN) is designed to identify and classify duplicate or similar bug reports. Three supervised machine learning methods—logistic regression, Naive Bayes, and decision tree—predict software defects using historical data. Model refactoring uses an advanced neural network to find functional errors in object-oriented software UML models [48]. The study advises using data science tools to understand multidimensional software design. The data is used to extrapolate and understand intricate architectural relationships.

IV. CONCLUSION

The application of machine learning in the field of software engineering has been the subject of substantial efforts undertaken by a number of writers. Their primary objective was to deliver unbiased evaluations. However, it can involve a degree of subjectivity. Moreover, it is important to highlight that this research study showcases the practicality of a Various machine learning techniques are applied throughout different stages of the software development life cycle (SDLC). The fact that successfully integrating machine learning algorithms into the software development process is a task that is extremely demanding is a significant discovery that has considerable implications. To summarise, this study highlights the necessity of efficient cooperation among researchers for the purpose of addressing the issues that are now being faced in the disciplines of machine learning (ML) and software engineering (SE).

REFERENCES

[1] L. E. Lwakatare, A. Raj, J. Bosch, H. H. Olsson, I. Crnkovic. A taxonomy of software engineering challenges for machine learning systems: An empirical investigation, in P. Kruchten, S. Fraser, F. Coallier (Eds.), Agile processes in software engineering and extreme programming. XP 2019. Lecture notes in business information processing, Vol. 355, Springer, Cham, 227-243 (2019). doi: 10.1007/978-3-030-19034-7_14

[2] M. Shehab, L. Abualigah, M. I. Jarrah, O. A. Alomari, M. S. Daoud MS. Artificial intelligence in software engineering and inverse. International Journal of Computer Integrated Manufacturing, 33(10-11), 1129-1144 (2020). doi: 10.1080/0951192X.2020.1780320

[3] V. H. Durelli, R. S. Durelli, S. S. Borges, A. T. Endo, M. M. Eler, D. R. Dias, M. P. Guimaraes. Machine learning applied to software testing: A systematic mapping study. IEEE Transactions on Reliability, 68(3), 1189-1212 (2019). doi: 10.1109/TR.2019.2892517

[4] F. Khomh, B. Adams, J. Cheng, M. Fokaefs, G. Antoniol. Software engineering for machine-learning applications: The road ahead. IEEE Software, 35(5), 81-84 (2018). doi: 10.1109/MS.2018.3571224

[5] N. Maneerat, P. Muenchaisri. Bad-smell prediction from software design model using machine learning techniques, in 2011 Eighth international joint conference on computer science and software engineering (JCSSE), 11-13 May 2011, Nakhonpathom, Thailand, 331-336 (2011). doi: 10.1109/JCSSE.2011.5930143

[6] P. Talele, R. Phalnikar. Software requirements classification and prioritisation using machine learning, in A. Joshi, M. Khosravy, N. Gupta (Eds.), Machine learning for predictive analysis. Lecture notes in networks and systems, Vol. 141, Springer, Singapore, 257-267 (2021). doi: 10.1007/978-981-15-7106-0_26

[7] J. Zou, L. Xu, W. Guo, M. Yan, D. Yang, X. Zhang. Which non-functional requirements do developers focuson? An empirical study on stack overflow using topic analysis, in 2015 IEEE/ACM 12th working conference on mining software repositories, 16-17 May 2015, Florence, Italy, 446-449 (2015). doi: 10.1109/MSR.2015.60

[8] A. Ahmad, K. Li, C. Feng, T. Sun. An empirical study on how iOS developers report quality Aspects on stack overflow. International Journal of Machine Learning and Computing, 8(5), 501-506 (2018).

[9] C. Treude, O. Barzilay, M. A. Storey. How do programmers ask and answer questions on the web? Nier track, in 2011 33rd International conference on software engineering (ICSE), 21-28 May 2011, Waikiki, Honolulu, HI, USA, 804-807 (2011). doi: 10.1145/1985793.1985907

[10] J. Zou, L. Xu, M. Yang, X. Zhang, D. Yang. Towards comprehending the non-functional requirements through developers' eyes: An exploration of stack overflow using topic analysis. Information and Software Technology, 84, 19-32 (2017). doi: 10.1016/j.infsof.2016.12.003

[11] Ahmad, C. Feng, K. Li, S. M. Asim, T. Sun. Toward empirically investigating non-functional requirements of iOS developers on stack overflow. IEEE Access, 7, 61145- 61169 (2019). doi: 10.1109/ACCESS.2019.2914429

[12] H. . Yin, D. Pfahl. A preliminary study on the suitability of stack overflow for open innovation in requirements engineering, in Proceedings of the 3rd international conference on communication and information processing, 24-26 November 2017, Tokyo, Japan, 45-49 (2017). doi: 10.1145/3162957.3162965

[13] R. Ferenc, A. Beszedes, L. Fulop, J. Lele. Design pattern mining enhanced by machine learning, in 21st IEEE international conference on software maintenance (ICSM'05), 26- 29 September 2005, Budapest, Hungary, 295-304 (2005). doi: 10.1109/ICSM.2005.40

[14] M. Zanoni, F. A. Fontana, F. Stella. On applying machine learning techniques for design pattern detection. Journal of Systems and Software, 103, 102-117 (2015). doi: 10.1016/j.jss.2015.01.037

[15] R. Selvarani, P. Mangayarkarasi. A dynamic optimization technique for redesigning OO software for reusability. ACM SIGSOFT Software Engineering Notes, 40(2), 1-6 (2015). doi: 10.1145/2735399.2735415

[16] R. Agashe, S. Iyer, L. Zettlemoyer. Juice: A large scale distantly supervised dataset for open domain context-based code generation (2019). doi: 10.48550/arXiv.1910.02216

[17] E. C. Shin, M. Allamanis, M. Brockschmidt, A. Polozov. Program synthesis and semantic parsing with learned code idioms, in 33rd Conference on neural information processing systems (NeurIPS 2019), Vancouver, Canada (2019).

[18] A. Takahashi, H. Shiina, N. Kobayashi. Automatic generation of program comments based on problem statements for computational thinking, in 2019 8th International congress on advanced applied informatics (IIAI-AAI), 07-11 July 2019, Toyama, Japan, 629-634 (2019). doi: 10.1109/IIAI-AAI.2019.00132

[19] Y. Shido, Y. Kobayashi, A. Yamamoto, A. Miyamoto, T. Matsumura. Automatic source code summarization with extended tree-lstm, in 2019 International joint conference on neural networks (IJCNN), 14-19 July 2019, Budapest, Hungary, 1-8 (2019). doi: 10.1109/IJCNN.2019.8851751

[20] Z. Zhu, Z. Xue, Z. Yuan. Automatic graphics program generation using attention-based hierarchical decoder, in C. Jawahar, H. Li, G. Mori, K. Schindler (Eds.), Computer vision – ACCV 2018. ACCV 2018. Lecture notes in computer science, Vol. 11366, Springer, Cham, 181-196 (2019). doi: 10.1007/978-3-030-20876-9_12

[21] H. Alaqail, S. Ahmed. Overview of software testing standard ISO/IEC/IEEE 29119. nternational Journal of Computer Science and Network Securit, 18(2), 112-116 (2018).

Mr. Vikas is working as Assistant Professor in Ajay Kumar Garg Engineering College. He is currently pursuing his Ph.D. from Birla Institute of Technology, Mesra, Ranchi. He is having more than 11 years of experience in teaching field and 1.2 years of experience in IT industry. His area of interest is in Networking and Machine Learning.