AI-Driven Software Testing: Reducing Bugs with Predictive Models

**Author:**

**Daniel Raj Jeevaguntala,**

**Affiliation:**

**Department of Health, Division of Operations and Systems, State of New York, USA**

**ABSTRACT:**

Conventional methods of quality assurance have been significantly altered as a result of the incorporation of capabilities related to artificial intelligence (AI) into software testing. With the help of predictive models, artificial intelligence-driven testing attempts to improve the identification of defects, simplify testing methods, and reduce the amount of manual work that is required thanks to the utilization of these models. The purpose of this systematic review is to assemble the existing methodology, instruments, and empirical data associated with the practical applications of artificial intelligence in software testing. In addition to this, the study highlights the significance of these programs in providing a reduction in the number of bugs and describes potential future research subjects.

1. **INTRODUCTION:**

Since it guarantees the delivery of dependable and high-quality products, the testing stage of the software development lifecycle is among the most crucial ones. Sometimes laborious and time-consuming, traditional testing techniques find it challenging to keep up with the ever growing complexity of contemporary software systems and the fast release cycles they experience. Artificial intelligence driven testing brings automation and intelligence into testing processes, so offering solutions to these challenges by means of mistake prediction, test case generation, and test execution optimization.

* 1. **AI-Driven Software Testing:**

Through the utilization of artificial intelligence (AI) and machine learning (ML) methodologies, AI-driven software testing is able to enhance the traditional testing approach [1]. AI has the ability to analyze enormous amounts of data, recognize trends, adjust to changing circumstances, and automatically test applications by mimicking the behavior of real-world users [2]. This is in contrast to the traditional method of relying solely on hand testers or pre-made scripts. This approach is far more effective in terms of testing accuracy, efficiency, and scalability.

Using predictive analytics, AI-driven testing provides one of the most significant benefits: it has the potential to discover defects at an earlier stage in the development cycle. AI is able to analyze previous problem reports, recognize patterns, and make recommendations for the areas of an application that are most likely to fail. While simultaneously increasing the dependability of software, this proactive method helps reduce the amount of time and money spent on development [3, 4].

Figure Key Features of AI driven testing Systems

Testing systems that are powered by artificial intelligence have the ability to further improve test automation by dynamically developing test cases, increasing test coverage, and hence reducing the number of duplicate test scripts [5]. When user interface changes occur, AI-driven testing may be able to self-heal test scripts, hence reducing the amount of downtime and manual work required. This is in contrast to traditional automated testing, which requires ongoing human intervention for maintenance. Artificial intelligence makes it easier for DevOps and Agile environments to be tested at all times by engaging with continuous integration and continuous delivery pipelines. Through the discovery of performance bottlenecks, it ensures a higher level of software quality, accelerates release cycles, and enhances the user experience [6]. Additionally, AI-driven testing can review user feedback in real time, which has the effect of improving the responsiveness of the application. On account of the ever-increasing complexity of modern applications, artificial intelligence-driven software testing is rapidly becoming an essential innovation in the field of software development and quality assurance [7].

Figure Benefits of integrating AI and software’s in bug detection

* 1. **Methods long-standing in use for quality control and bug detection:**

Software testing has always been done in the course of existence using manual and automated approaches. Manual testing is the method whereby human testers complete pre-defined test cases to find flaws. Time-consuming, error-prone, and resource-intensive this kind of testing can be is. Conversely, automated testing runs test cases in a more effective way using scripts and tools; however, it depends on predefined patterns and stationary scenarios. Though they have been basic, traditional methods have several restrictions. This is particularly true in the context of always shifting and dynamic software environments, where conventional testing could not be able to identify new or evolving flaws. Furthermore challenging to scale testing initiatives or generate consistent results across large codebases or frequent release cycles is the fact that manual testing depends on human interaction [8].

1. **METHODOLOGY:**

According to the principles established by the Preferred Reporting Items for Systematic Reviews and Meta-Analyses (PRISMA), this review is being conducted. A comprehensive search was conducted across a wide variety of databases, including IEEE Xplore, ACM Digital Library, ScienceDirect, and arXiv, with the primary focus being on papers published between the years 2018 and 2025. The following are some of the terms that were utilized: "AI in software testing," "predictive models," "machine learning in testing," and "bug prediction." A total of 55 relevant studies were selected for further investigation after being evaluated based on inclusion and exclusion criteria.

* 1. **Search Strategy:**

We carefully reviewed the literature using electronic databases including: IEEE Investigate, Online ACM Library, Science Direct, arrXiv , Google Scholar and Researchgate. Keywords or terms used for this survey includes, artificial intelligence in software testing, Predictive Models for Bug reduction, Matrix Learning for Defection Prediction, AI-powered testing automaton, Defect Prediction Software engineering. We includes all studies from 2018 till April 2025 in this study [9-64].

* 1. **Inclusion and Exclusion Criteria:**

This study Includes research that make use of artificial intelligence in software testing, prioritize the identification of bugs, the optimization of tests and the prediction of defects. Peer review of articles within a journal or conference. This study excludes all the studies excluding predictive modeling or artificial intelligence, grey literature—that which comes from blogs and magazines and duplicates or unavailability of complete texts

Figure Prisma Flowchart of methodology and studies includes

1. **Results:**
	1. **Distribution Of Studies By Focus Area:**

During the course of this systematic review, the majority of the papers that were under consideration focused on defect prediction through the utilization of artificial intelligence and machine learning models. These investigations mostly utilized neural networks, support vector machines (SVM), and Random Forest as classification algorithms in order to identify software systems that have modules that are prone to defects. With the objective of automating or prioritizing test cases through the utilization of artificial intelligence in order to accelerate the testing cycle and enhance productivity, more than 27 percent of the selected research focused on test optimization. In the meantime, 18 percent of the research concentrated on anomaly detection through the utilization of unsupervised learning methods in order to identify anomalies in software behavior that could potentially indicate latent problems. Despite the fact that predictive modeling is still the most common application of artificial intelligence in software testing, this distribution highlights the fact that optimization and anomaly detection are becoming regions of interest for further development [1, 9, 12, 19, 20, 65-69].



Figure 4 Distribution Of Studies By Focus Area

* 1. **Key Findings:**

The findings of this comprehensive research shed light on the numerous advantages that can be gained from automating software testing with artificial intelligence. With research showing that predicted accuracies can average up to 85% across a number of different machine learning models, defect prediction accuracy emerged as the most successful result, as can be seen in the bar graph. Using techniques that are based on artificial intelligence for prioritizing and optimizing test cases, the amount of time it takes to execute tests decreased by an average of 35 percent. In addition, the use of AI approaches helped to reduce the number of false positives that occurred during the bug discovery operations, which ultimately led to more accurate testing findings. In addition, the automation of repetitive tasks led to a reduction of forty percent in the amount of manual testing that was performed, which enabled human testers to devote more of their attention to testing situations that were more exploratory and difficult. These findings demonstrate the growing sophistication of systems that utilize artificial intelligence in software testing, which therefore offers clear benefits in terms of accuracy, efficiency, and resource economy.

* 1. **Performance by AI Technique:**

The outcomes of this systematic research make it abundantly evident that artificial intelligence-driven software testing has a number of attractive advantages. The accuracy of defect prediction emerged as the most significant outcome displayed in the bar graph. This is due to the fact that studies have demonstrated that anticipated accuracies average up to 85% across a variety of machine learning models. The average amount of time required to execute a test decreased by 35 percent, mostly as a result of the implementation of AI-based test case prioritizing and optimization techniques. Artificial intelligence tools also helped to reduce the number of false positives that occurred during the process of discovering issues, which resulted in more accurate testing outcomes. In addition, the automation of repetitive processes resulted in a reduction of manual testing activities by forty percent, which enabled human testers to devote more of their attention to testing environments that were more exploratory and difficult. The findings presented here shed light on the increasing sophistication of programs that utilize artificial intelligence in software testing. As a result, these applications provide several distinct advantages in terms of accuracy, efficiency, and resource economy.

**Discussions:**

The empirical findings of this comprehensive investigation reveal that AI-driven approaches have significantly improved a number of elements of software testing, particularly in the areas of defect prediction, test optimization, and anomaly detection. Among the research that was looked into, machine learning models such as Random Forest and Support Vector Machines demonstrated the highest level of accuracy when it came to forecasting defect-prone components. Random Forest was able to achieve a prediction accuracy of up to 88% [70]. These findings shed light on the fact that ensemble techniques are capable of effectively managing difficult, high-dimensional software datasets. Deep neural networks were notably useful for occupations such as automated test case development, where their ability to express nonlinear patterns proved to be highly important. Despite the fact that their prediction performance around 81% is significantly lower, deep neural networks were particularly effective for these jobs [71]. In addition, clustering approaches such as K-Means proved to be beneficial in unsupervised scenarios for spotting irregularities in test outputs that suggested latent software flaws. This was the case despite the fact that their accuracy was lowered by 76%. In addition to improving prediction performance, the implementation of artificial intelligence techniques resulted in a significant reduction in the amount of time required to carry out tests by at least 30–40% [72]. The most fault-revealing test cases were discovered early on in the testing cycle using test prioritizing models, which resulted in a reduction in the amount of time each test was executed. In addition, around twenty-five percent contributed to the reduction of false positive rates in bug discovery, which meant that the accuracy of test results was improved, and human testers were relieved of some of their responsibilities. The reduction of manual testing efforts was another obvious advantage of automation technologies [29]. As a result of the reduction in manual workloads, up to forty percent of human testers might be freed up to focus on exploratory testing and creative problem-solving. However, empirical studies has demonstrated that the quality and amount of the training datasets have a substantial impact on the performance of the model. Problems such as data imbalance have a detrimental impact on the accuracy of defect prediction. Additionally, despite the fact that artificial intelligence models brought about improvements in accuracy and efficiency, as software technologies advanced, they faced challenges in terms of the explainability and adaptability of the models over time. These findings highlight the significant potential of AI-driven software testing in terms of enhancing quality assurance methods. Additionally, they highlight the necessity of continuous research on model transparency, dataset curation, and techniques for maintaining model relevance throughout the lifetime of dynamic software [73].



Figure Overview of review

**CONCLUSIONS:**

Emerging as a transforming tool to solve long-standing issues with manual, time-consuming, and often error-prone quality assurance procedures is artificial intelligence (AI) integration into software testing. Particularly via predictive models, this systematic analysis shows that AI-driven testing provides notable improvements in early software fault discovery, cycle optimization, and manual workload reduction. Exceptional potential has been shown by machine learning techniques such Random Forest, Support Vector Machines (SVMs), and Deep Neural Networks, which exhibit high degrees of predicted accuracy over many datasets and application settings. These models help software teams to predict defect-prone modules with amazing accuracy, so enabling preemptive interventions that raise general dependability and quality of software. Beyond defect prediction, artificial intelligence methods have greatly raised operational efficiency. Research presented in this review showed that test case prioritizing models decreased test execution times by up to 40%, therefore shortening development cycles and allowing faster delivery of software products without sacrificing quality. Likewise, the decrease of false positives by about 25% has improved the credibility of automated testing findings, therefore saving time and effort needed for pointless fault searches. By means of intelligent systems, the automation of routine and repetitive testing activities has resulted in a notable 40% reduction in manual testing efforts, so enabling the reallocation of skilled testers toward exploratory testing, usability evaluations, and other higher-order activities benefiting from human judgment. The results also show, meanwhile, that artificial intelligence-driven software testing is not without difficulties even if the obvious advantages abound. Not usually available in real-world testing situations, large, high-quality, well-annotated datasets are absolutely essential for the development of machine learning models. Furthermore, many artificial intelligence algorithms—especially deep learning models—have black-box character that begs questions about explainability and responsibility—two important factors when AI models are used in applications with safety-critical relevance. Moreover, software systems are dynamic, always developing with new needs, tools, and designs. Predictive models built on historical data may thus lose performance over time if they are not retrained or adjusted to represent the changing software environment, a phenomena also referred to as idea drift. Overcoming these obstacles must be the main emphasis of future studies if we are to completely realize the possibilities of artificial intelligence in software testing. Encouragement of more general acceptance and confidence depends on developing explainable artificial intelligence (XAI) models with openness into prediction mechanisms. While maintaining data privacy, methods such federated learning and transfer learning should help models to generalize more across projects and companies. Furthermore, the development of standardized, publicly accessible benchmark datasets for AI-based software testing will help to enable more consistent and comparable assessments of several methodologies. Combining the strengths of machine intelligence with human knowledge can also help to guarantee that AI models are monitored and rectified by expert testers by means of human-in-the-loop technologies. Ultimately, carefully considered and ethically applied artificial intelligence-driven software testing has the ability to transform the quality assurance process. Apart from enhancing operational efficiency and problem identification, it helps software engineering teams to produce better products faster and more economically. AI will progressively become an invaluable friend in negotiating the complexity of modern software development as research advances and technology mature, bringing in a new era of intelligent, predictive, and adaptive software testing approaches.

**References:**

1. Ahmed, H.A., N.Z. Bawany, and J.A. Shamsi, *Capbug-A framework for automatic bug categorization and prioritization using NLP and machine learning algorithms.* IEEE Access, 2021. **9**.

2. Ateya, H.A.B. and S.M. Baneamoon, *Software bug prediction using static analysis with abstract syntax trees.* International Journal of Engineering and Artificial Intelligence, 2020. **1**.

3. Choetkiertikul, M., et al., *Automatically recommending components for issue reports using deep learning.* Empirical Software Engineering, 2021. **26**.

4. Du, X., et al., *Cross-project bug type prediction based on transfer learning.* Software Quality Journal, 2020. **28**.

5. Ferenc, R., et al., *An automatically created novel bug dataset and its validation in bug prediction.* Journal of Systems and Software, 2020. **169**.

6. Giray, G., et al., *On the use of deep learning in software defect prediction.* Journal of Systems and Software, 2023. **195**.

7. Gray, D., et al., *Reflections on the NASA MDP data sets.* IET Software, 2012. **6**.

8. Peterson, B., *Adaptive AI Models in Software Testing: Leveraging Predictive Analytics to Automate Bug Detection and Quality Assurance.* 2025.

9. Afric, P., et al., *Empirical Study: How Issue Classification Influences Software Defect Prediction.* IEEE Access, 2023. **11**.

10. Choetkiertikul, M., et al., *Automatically recommending components for issue reports using deep learning.* Empir Softw Eng, 2021. **26**.

11. Fang, S., et al., *Effective Prediction of Bug-Fixing Priority via Weighted Graph Convolutional Networks.* IEEE Trans Reliab, 2021. **70**.

12. Giray, G., et al., *On the use of deep learning in software defect prediction.* J Syst Softw, 2023. **195**.

13. Gupta, D.L. and K. Saxena, *Software bug prediction using object-oriented metrics.* Sādhanā, 2017. **42**.

14. Hammouri, A., et al., *Software bug prediction using machine learning approach.* Int J Adv Comput Sci Appl, 2018. **9**.

15. Iqbal, A., et al., *Performance analysis of machine learning techniques on software defect prediction using NASA datasets.* Int J Adv Comput Sci Appl, 2019. **10**.

16. Khan, F., et al., *Hyper-parameter optimization of classifiers, using an artificial immune network and its application to software bug prediction.* IEEE Access, 2020. **8**.

17. Mohammadi-Balani, A., et al., *Golden eagle optimizer: A nature-inspired metaheuristic algorithm.* Comput Ind Eng, 2021. **152**.

18. Nucci, D., et al., *A developer centered bug prediction model.* IEEE Trans Softw Eng, 2017. **44**.

19. Panda, R.R. and N.K. Nagwani, *An intuitionistic fuzzy representation based software bug severity prediction approach for imbalanced severity classes.* Eng Appl Artif Intell, 2023. **122**.

20. Qiao, L., et al., *Deep learning based software defect prediction.* Neurocomputing, 2020. **385**.

21. *Cynthia ST, Roy B, Mondal D (2022) Feature transformation for improved software bug detection models. In 15th Innovations in Software Engineering Conference, pp. 1–10.*

22. *Wang Z, Tong W, Li P, Ye G, Chen H, Gong X, Tang Z (2023) BugPre: an intelligent software version-to-version bug prediction system using graph convolutional neural networks. Complex Intell Syst 9(4):3835–3855.*

23. *Sivapurnima S, Manjula D (2023) Adaptive deep learning model for software bug detection and classification. Comput Syst Sci Eng 45(2):1234–1248.*

24. *Chouhan SS, Rathore SS (2021) Generative adversarial networks-based imbalance learning in software aging-related bug prediction. IEEE Trans Reliab 70(2):626–642.*

25. *Gupta A, Sharma M, Srivastava A (2023) Group Counselling Optimization-derived Software Bug Prediction Framework by Optimal Feature Selection-based Dimensionality Reduction. In 2023 13th International Conference on Cloud Computing, Data Science & Engineering (Confluence) (pp. 636–642). IEEE.*

26. *Gupta A, Sharma M, Srivastava A (2023) A Novel Dimensionality Reduction-based Software Bug Prediction using a Bat-Inspired Algorithm. In 2023 13th International Conference on Cloud Computing, Data Science & Engineering (Confluence) (pp. 278–285). IEEE.*

27. Xu, Z., et al., *Software defect prediction based on kernel PCA and weighted extreme learning machine.* Inf Softw Technol, 2019. **106**.

28. Wahono, R.S., *A systematic literature review of software defect prediction.* J Softw Eng, 2015. **1**.

29. Turabieh, H., M. Mafarja, and X. Li, *Iterated feature selection algorithms with layered recurrent neural network for software fault prediction.* Expert Syst Appl, 2019. **122**.

30. Abdullahi, M., *Detecting cybersecurity attacks in internet of things using artificial intelligence methods: a systematic literature review.* Electronics, 2022. **11**.

31. Ahmed, H.U., *Technology developments and impacts of connected and autonomous vehicles: an overview.* Smart Cities, 2022. **5**.

32. Ahmed, S. and A. Alhumam, *Unified computational modelling for healthcare device security assessment.* Comput. Syst. Sci. Eng., 2021. **37**.

33. Akhmetov, B., *Automation of information security risk assessment.* Int. J. Electr. Telecommun., 2022. **68**.

34. *Al-Turkistani, H.F. and A. AlFaadhel. 2021. "Cyber resiliency in the context of cloud computing through cyber risk assessment". In 2021 1st International Conference on Artificial Intelligence and Data Analytics (CAIDA). IEEE.*

35. *Alonge, C.Y., et al. 2020. "Information asset classification and labelling model using fuzzy approach for effective security risk assessment". In 2020 International Conference in Mathematics, Computer Engineering and Computer Science (ICMCECS). IEEE.*

36. *Alperin, K., et al. 2019. "Risk prioritization by leveraging latent vulnerability features in a contested environment". In Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security.*

37. *Alperin, K.B., A.B. Wollaber, and S.R. Gomez. 2020. "Improving interpretability for cyber vulnerability assessment using focus and context visualizations". In 2020 IEEE Symposium on Visualization for Cyber Security (VizSec). IEEE.*

38. *Anisetti, M., C.A. Ardagna, and N. Bena. 2023. "Continuous Certification of Non-functional Properties Across System Changes". In International Conference on Service-Oriented Computing. Springer.*

39. *Bahja, M. 2020. "Natural language processing applications in business". E-Business-higher education and intelligence applications.*

40. *Bettaieb, S., et al. 2019. "Decision support for security-control identification using machine learning". In International Working Conference on Requirements Engineering: Foundation for Software Quality. Springer.*

41. *Bo, T., et al. 2019. "Tom: A threat operating model for early warning of cyber security threats". In Advanced Data Mining and Applications: 15th International Conference, ADMA 2019, Dalian, China, November 21–23, 2019, Proceedings 15. Springer.*

42. *Boyce, J. and D. Jennings. 2002. "Information assurance: Managing organizational IT security risks". volume: Butterworth-Heinemann.*

43. *Deimling, F. and M. Fazzolari. 2023. "AMOE: A Tool to Automatically Extract and Assess Organizational Evidence for Continuous Cloud Audit". In IFIP Annual Conference on Data and Applications Security and Privacy. Springer.*

44. *Dike, H.U., et al. 2018. "Unsupervised learning based on artificial neural network: A review". In 2018 IEEE International Conference on Cyborg and Bionic Systems (CBS). IEEE.*

45. *Ferdinand, M.R., S. Mandala, and D. Oktaria. 2021. "Host Vulnerability Analysis Using Supervised Learning Based on Port Response". In 2021 International Conference on Intelligent Cybernetics Technology & Applications (ICICyTA). IEEE.*

46. *Ghanem, M.C. and T.M. Chen. 2018. "Reinforcement learning for intelligent penetration testing". In 2018 Second World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4). IEEE.*

47. *Grieco, G. and A. Dinaburg. 2018. "Toward smarter vulnerability discovery using machine learning". In Proceedings of the 11th ACM Workshop on Artificial Intelligence and Security.*

48. *Hecker, A. and M. Riguidel. 2009. "On the operational security assurance evaluation of networked IT systems". In Smart Spaces and Next Generation Wired/Wireless Networking: 9th International Conference, NEW2AN 2009 and Second Conference on Smart Spaces, ruSMART 2009, St. Petersburg, Russia, September 15–18, 2009. Proceedings. Springer.*

49. *Hayrapetian, A. and R. Raje. 2018. "Empirically analyzing and evaluating security features in software requirements". In Proceedings of the 11th Innovations in Software Engineering Conference.*

50. *Hosam, O. 2022. "Intelligent risk management using artificial intelligence". In 2022 Advances in Science and Engineering Technology International Conferences (ASET). IEEE.*

51. *Hsieh, C.-H., et al. 2015. "Cyber security risk assessment using an interpretable evolutionary fuzzy scoring system". In 2015 International Carnahan Conference on Security Technology (ICCST). IEEE.*

52. Zhou, F., *Fingerprinting IIoT devices through machine learning techniques.* J. Signal Proc. Syst., 2021. **93**.

53. Zeng, Z., *Licality—likelihood and criticality: Vulnerability risk prioritization through logical reasoning and deep learning.* IEEE Trans. Netw. Serv. Manage., 2021. **19**.

54. Yüksel, A.S., *Implementation of a web-based service for mobile application risk assessment.* Turk. J. Electr. Eng. Comput. Sci., 2017. **25**.

55. Yi, J. and X. Liu, *Deep reinforcement learning for intelligent penetration testing path design.* Appl. Sci., 2023. **13**.

56. Yadav, S.B. and T. Dong, *A comprehensive method to assess work system security risk.* Commun. Assoc. Inform. Syst., 2014. **34**.

57. Xiao, J., *Black-box attack-based security evaluation framework for credit card fraud detection models.* INFORMS J. Comput., 2023. **35**.

58. Williams, I., *An automated security concerns recommender based on use case specification ontology.* Auto. Softw. Eng., 2022. **29**.

59. Wiafe, I., *Artificial intelligence for cybersecurity: a systematic mapping of literature.* IEEE Access, 2020. **8**.

60. Wen, S.F. and B. Katt, *Exploring the role of assurance context in system security assurance evaluation: a conceptual model.* Inform. Comput. Security, 2024. **32**.

61. Singh, V.K., *The journal coverage of web of science, scopus and dimensions: a comparative analysis.* Scientometrics, 2021. **126**.

62. Silvestri, S., *Cyber threat assessment and management for securing healthcare ecosystems using natural language processing.* Int. J. Inf. Secur., 2024. **23**.

63. Shi, F., *XLNet-based prediction model for CVSS metric values.* Appl. Sci., 2022. **12**.

64. Sharma, K. and A. Mukhopadhyay, *Cyber-risk management framework for online gaming firms: an artificial neural network approach.* Inf. Syst. Front., 2023. **25**.

65. Hammouri, A., et al., *Software bug prediction using machine learning approach.* International Journal of Advanced Computer Science and Applications, 2018. **9**.

66. Juneja, K., *A fuzzy-filtered neuro-fuzzy framework for software fault prediction for inter-version and inter-project evaluation.* Applied Soft Computing, 2019. **77**.

67. Huang, Q., et al., *Automating intention mining.* IEEE Transactions on Software Engineering, 2020. **46**.

68. Xuewu, Z.H.A.O., et al., *Artificial jellyfish search optimization algorithm for human brain functional parcellation.* Journal of Frontiers of Computer Science & Technology, 2022. **16**.

69. Johnson, F., et al., *Optimized ensemble machine learning model for software bugs prediction.* Innovations Syst Softw Eng, 2023. **19**.

70. Ramay, W.Y., et al., *Deep neural network-based severity prediction of bug reports.* IEEE Access, 2019. **7**.

71. Rathore, S.S. and S. Kumar, *A decision tree logic based recommendation system to select software fault prediction techniques.* Computing, 2017. **99**.

72. Rathore, S.S. and S. Kumar, *Linear and non-linear heterogeneous ensemble methods to predict the number of faults in software systems.* Knowl-Based Syst, 2017. **119**.

73. Sherstinsky, A., *Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network.* Physica D, 2020. **404**.