engineering proceedings

Proceeding Paper

Automated Testing with Machine Learning Frameworks: A Critical Analysis [†]

Sana Fatima [*], **Bisma Mansoor, Laiba Ovais, Sajid Ali Sadruddin and Syed Aun Hashmi**

Department of Software Engineering, NED University of Engineering & Technology, Karachi 75270, Pakistan; mansoor4102449@cloud.neduet.edu.pk (B.M.); laiba4101071@cloud.neduet.edu.pk (L.O.); sadruddin4108747@cloud.neduet.edu.pk (S.A.S.); hashmi4102782@cloud.neduet.edu.pk (S.A.H.)
* Correspondence: sanafatima@cloud.neduet.edu.pk
† Presented at the 7th International Electrical Engineering Conference, Karachi, Pakistan, 25–26 March 2022.

Abstract: As software systems are becoming more and more complex and standard testing practices are exhausting, we need smart solutions to reduce the time, efforts and resources spent on software testing. The aim of this paper was to critically analyze machine learning (ML) frameworks related to software automation. We measured the performance of testing tools on the basis of the manual labor (effort) required, in addition to the test performance, accuracy or error rate, scope, time required and prerequisite knowledge requirements. These factors play a vital role to ensure ML frameworks with automation software can produce great results and hence improve software quality.

Keywords: machine learning; software quality; software testing; automation; performance; manual labor; accuracy; test scope; time required; prerequisite knowledge requirement

check for updates

Citation: Fatima, S.; Mansoor, B.; Ovais, L.; Sadruddin, S.A.; Hashmi, S.A. Automated Testing with Machine Learning Frameworks: A Critical Analysis. *Eng. Proc.* **2022**, *20*, 12. https://doi.org/10.3390/engproc2022020012

Academic Editor: Saad Ahmed Qazi

Published: 28 July 2022

1. Introduction

One of the most important parts of the software development cycle is testing. We can see many companies invest almost half of their resources in testing. Many researchers are trying to improve their software quality testing methods and we made significant progress from redundant manual testing to automation test cases. Automation is a major breakthrough in software testing [1]. Automation testing allows developers to write test cases that can be executed on their own and provide instant feedback for what happens on some inputs. The most important benefit that automation testing offers is a decreased testing time and effort. Even with this huge breakthrough, there is still a huge room for improvement. Automation testing, though considered automatic, still requires a lot of human effort in selecting the right tool and skilled resources.

However, in the era of the ever-growing field of artificial intelligence, we should expect software to behave intelligently [2]. Our paper highlights the importance and use cases of artificial intelligence and machine learning frameworks in making automation testing or testing in general better [3]. ML frameworks will make use of such algorithms that make automation software somewhat intelligent. The goal of our research is to categorize the use of ML frameworks in software testing and how they have an edge over normal automation testing. Our proposed framework will not only reduce the time and effort spent on testing but will also help to increase the quality of the software being tested. In a literature review, we reviewed some previous works to understand the current solutions. Then, in the methodology, we described the ML framework we are analyzing. The case study about TechM's analytics tool was performed so that we have good grounds for comparison. Afterwards, analysis was performed on the results we gathered.

2. Literature Review

In this paper [1], the researchers explained some critical factors that affect the software's cost, quality and time to market and how automation can minimize these factors.

They discovered that there was a need for a solution that can calculate the effects of test automation or benefits in terms of the aforementioned factors. They calculated the efforts and price for both manual and automation software testing. Automation testing took less time and effort, decreased the time to market but was a bit costly but since efforts were reduced by a big margin and the cost difference was not that significant.

In [3], some machine learning frameworks were discussed that could be used in automated testing tools. They proposed a new machine learning framework to increase the performance of software testing tools. They further analyzed and classified some machine learning algorithms according to the part of testing that they can be utilized in. It was concluded that, by using their proposed ML methods, researchers can solve problems in automation software testing.

The researchers in [4] proposed deep learning techniques to predict defect occurrence and changes that can lead to more defects. They followed J. Han and M. Kamber [5] research and further improved the results. They used the dataset from six open source projects with 137,417 changes. The first task was to find the optimum features to apply deep learning algorithms. A logistic regression [6] algorithm was used. They constructed the model in four phases. First, they cleaned and labeled the dataset and the second phase was data processing. In the third phase of feature integration, a Deep Belief Network [7] was constructed. In the final phase, the classifier was constructed and predictions were made. The approach used in this paper was better than that in [5] because the average recall was 69% and the F1 score was 45%. In conclusion, their model can identify more than 50% defective changes by only viewing 20% of the total source code.

In [8], the researcher conducted a study to reduce the amount of test cases needed for software testing using artificial neural networks. The first section covers non-neural network approaches. The next section discusses neural network-based software testing approaches. The method to reduce test cases was input–output analysis with ANNs. The model outputted the list of test cases. The proposed method lies in the black box testing category and does not require any source code.

The researchers in [9] discussed some machine learning algorithms with their usage in software testing and how they can improve bug exploration. First, they analyzed the existing conventional approaches of software testing, manual testing and their impact on software quality. Then, they analyzed previous research wherein some ML approaches were used. This paper outlined how ML can be used in software testing processes in future testing tools.

3. Methodology

3.1. Modernization of Automation Testing

In order to revolutionize the field of automation testing, the use of artificial intelligence was deemed necessary. There are various major mechanisms that can be followed for the purpose of training a machine, namely supervised learning, unsupervised learning, deep learning (use of neural network) and reinforcement learning [10].

3.2. Use of Machine Learning

According to surveys, the field of automation testing will be deeply impacted by the employment of machine learning. By using machine learning in testing, we will allow our systems to learn from experience, i.e., the system would be able to assess the facts and data presented to it, build and run test cases on the said data and then study from the result of the test cases. This whole procedure would in turn effectively enhance the testing cycle. Machine learning helps the system to develop and provide more precise results in less time.

In this particular approach supervised machine learning, a set of data are fed to the system as training examples. The system internalizes this information and then predicts or classifies new data. The steps that are followed in these approaches are [10]:

1. Identify your data sources;
2. Train the machine to identify correlations;

3. Review predictions;
4. Find algorithms with sufficient accuracy;
5. Apply the predictions to new data;
6. Indicate the expected prediction accuracy;
7. Learn and evolve.

The other approach is unsupervised machine learning. Neural networks fall under this category and are often used in systems. Neural networks are based on the functioning of the actual human brain and are algorithms that study and generalize from abstract concepts. They use a set of variables that can be adjusted form the learning procedure until the required level of accuracy is achieved.

3.3. Implementing Machine Learning

Predictive analysis, based on supervised learning, makes use of the results that are achieved by the system by changing the algorithms based on those results. Using this approach, we can implement its techniques for the purpose of risk assessment. (Risk assessment is used to comprehend and diminish the risks involved). For risk assessment, the observed data points related to the impact of failure and the likelihood of failure are gathered. Using the supervised learning, these points are used to train the system. By thus training the system, new data can be subjected to these algorithms to reach accurate results. In an example [10], the risk assessment grid was used to represent the results collected devoid of the use of supervised learning and they all converged at the same column in the grid. It was not an efficient solution. After the use of predictive analytics, the following grid (shown in Figure 1) was obtained:

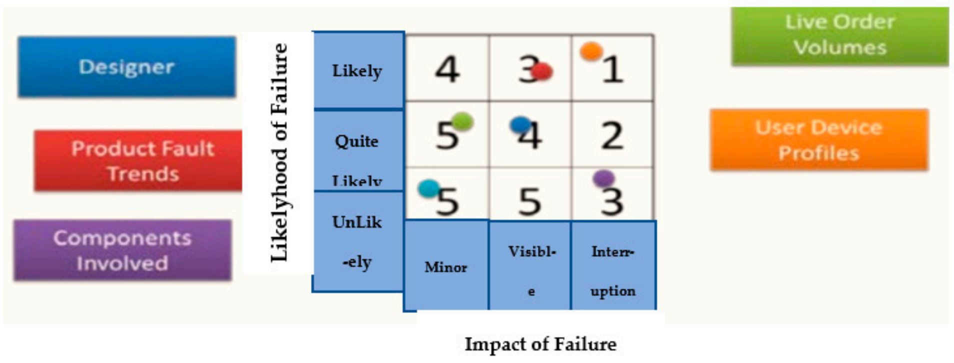

Figure 1. X3 Risk Assessment Grid.

3.4. Case Study

A test analytics [10] platform was developed by an Indian multinational company Tech Mahindra which has been integrated with their proprietary tool eConvergence. It uses machine learning to extract meaningful information from data collected from testing tools and helps to find the causes of defects and bugs by reducing the time for STLC. This test analytics tool captures analytic feeds from data regarding defects and tests and proceeds towards the defect prediction model which uses machine learning algorithms and helps predict the foresights in order to improve the business outcomes by analyzing the historical test practices. Figure 2 gives a clear demonstration of the working of TechM's analytics tool. Data from testing tools are transferred to the test reporting dashboard for aggregation and then proceed to the test analytics platform for predictive analysis based on historical data. In the last step, the tests are prioritized on the basis of the risk of their failure and predictions are made which are discussed in the following section.

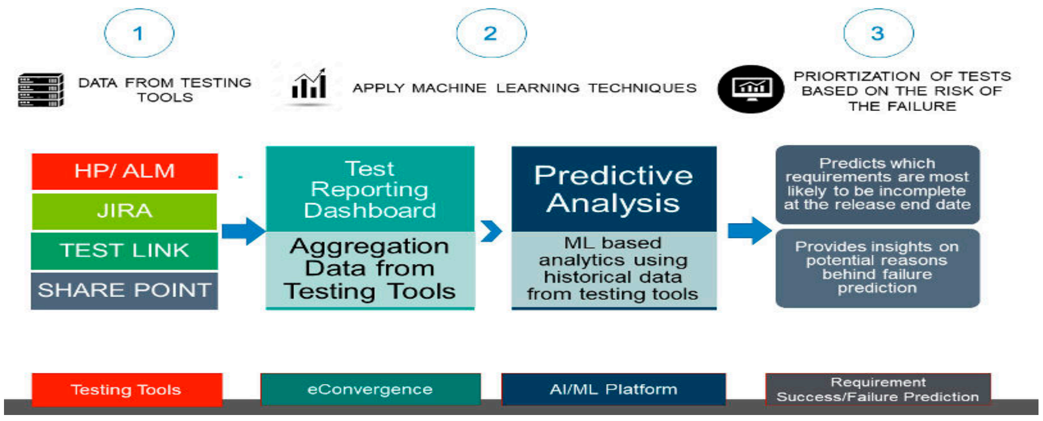
1
DATA FROM TESTING TOOLS
2
APPLY MACHINE LEARNING TECHNIQUES
3
PRIORTIZATION OF TESTS BASED ON THE RISK OF THE FAILURE
HP/ ALM
JIRA
TEST LINK
SHARE POINT
Test Reporting Dashboard
Aggregation Data from Testing Tools
Predictive Analysis
ML based analytics using historical data from testing tools
Predicts which requirements are most likely to be incomplete at the release end date
Provides insights on potential reasons behind failure prediction
Testing Tools
eConvergence
AI/ML Platform
Requirement Success/Failure Prediction

Figure 2. Diagrammatic view of TechM's ML-based tool.

3.5. Comparing Conventional and ML-Based Testing Tools

The comparison (Table 1) is performed by using six standard attributes, i.e., manual labor, test performance, error inaccuracy, test scope, time invested and prerequisite programming skills.

Table 1. Conventional and ML-based automation: a comparison.

Criterion	Conventional Test Automation	ML-Based Test Automation
Manual labor	More involvement of manual labor.	Less amount of manual labor required.
Test performances	Needs to be modified through proactive efforts.	Exhibits the property of self-healing.
Error and inaccuracy	Because of the high involvement of man power, conventional test automation is prone to a high number of errors and inaccuracies.	As the machine replaces most of the human efforts in ML-based automation, it is less likely to produce errors or in accuracies.
Test scope	The aspects or requirement in the software needs to test through active endeavor.	During the process of auto testing, novel aspects of software under consideration may be analyzed.
Time invested	In conventional automation, even small changes in the UI needs to be adjusted in the script as well.	As ML adjusts scripts according to the variation in the applications, a large amount of time does not need to be wasted in scripting.
Prerequisite programming skills	Conventional test automation demands prerequisite technical skills and knowledge.	Most ML-based tools can be used with sub-par technical knowledge or skills.

4. Analysis

Using machine learning in software test automation, software teams are now able to more effectively maintain and improve the quality of large software systems with a reduced cost and human efforts. With ML testing tools, the time required for a software testing life cycle can be reduced by half without compromising the quality of the software [10] by making predictions from historical data and enabling the testing and delivery teams to schedule reviews and walkthroughs accordingly [11]. These tools encompass self-adjusting scripts and self-healing properties which lessen the need for human intervention, thus making the procedure easier for people. It is observed that ML-based tools, due to their characteristic of learning from understanding, experience fewer inaccuracies and errors in their run. ML-based tools may not be a perfect solution but they surely have an edge over the conventional automation tools.

5. Limitations

Since artificial intelligence does offer promising results, it only concentrates on solving a specific testing problem; this means that there is no readily available overall toolkit for testing engineers to use. Hence, the complete adaptation of artificial intelligence in organizations becomes difficult; it is now inevitable that AI will have a substantial impact on the field of automation testing. We expect that new methodologies and approaches from an AI standpoint will emerge to tackle the task of software testing.

6. Conclusions and Future Work

The primary purpose of this study was to upsurge the awareness about the prospective benefits of AI in the field of software test automation. The research clearly concludes that with AI, the production and release of projects will be much quicker and efficient as it will allow tests to be developed faster and errors be discovered sooner. If and when code modifications are made to the application, AI/ML will be able to estimate the likelihood of a build failing. By learning through our procedures and tests, AI will optimize and further predict problematic areas requiring further attention from a tester. To conclude, the development and use of test automations backed by AI is an ideal way forward for everyone.

Our major effort for future research will be to examine and categorize all existing work in the field of machine learning and software testing. The information obtained throughout the categorization process will assist future researchers and engineers in developing a particular set of guidelines for using ML techniques in the software testing process.

Author Contributions: Conceptualization was performed by B.M.; methodology, L.O. and B.M.; investigation, S.A.H. and S.F.; writing—original draft preparation, S.A.H.; writing—review and editing, S.A.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Divya, K.; Mishra, K.K. The Impacts of Test Automation on Software's Cost, Quality and Time to Market. *Procedia Comput. Sci.* **2016**, *79*, 8–15.
2. Mitchell, T.M. *Machine Learning*; McGraw Hill: New York, NY, USA, 1997; ISBN 0070428077.
3. Jungho, K.; Ryu, Y.; Woo, J.; Hyun-Jeong, S.; Jin-Hee, S. Machine Learning Frameworks for Automated Software Testing Tools: A Study. *Int. J. Contents* **2017**, *13*, 38–44.
4. Yang, X.; David, L.; Xia, X.; Zhang, Y.; Sun, J. Deep Learning for Just-In Time Defect Prediction. In Proceedings of the IEEE International Conference on Software Quality, Reliability and Security, QRS, Vancouver, BC, Canada, 3–5 August 2015; pp. 17–26.
5. Kamei, Y.; Shihab, E.; Adams, B.; Hassan, A.E.; Mockus, A.; Sinha, A.; Ubayashi, N. A large-scale empirical study of just-in-time quality assurance. *TSE* **2013**, *39*, 757–773. [CrossRef]
6. Han, J.; Kamber, M. *Data Mining: Concepts and Techniques*, 2nd ed.; Morgan Kaufmann Publishers: Burlington, MA, USA, 2006.
7. Hinton, G.; Osindero, S.; Teh, Y.-W. A fast learning algorithm for deep belief nets. *Neural Comput.* **2006**, *18*, 1527–1554. [CrossRef] [PubMed]
8. Prachi, S.; Mark, L.; Abraham, K. Test Set Generation and Reduction with Neural Networks. *Artif. Intell. Methods Softw. Test.* **2004**, 101–132. [CrossRef]
9. Sumit, M.; Subhankar, M. Usage of Machine Learning in Software Testing. In *Automated Software Engineering: A Deep Learning-Based Approach*; Springer: Berlin, Germany, 2020. [CrossRef]

10. Chakraborty, S. How to use AI/ML Systems to Revolutionize Testing And Test Automation? *Int. J. Sci. Eng. Res.* **2019**, *10*, 290.
11. UTOR. AI and ML in Testing: How to Make Your Test Automation more Intelligent—UTOR. 2021. Available online: https://utor.com/topic/ai-and-ml-in-software-testing (accessed on 1 August 2021).