A Survey on Software Testing Automation using Machine Learning Techniques

Mustafa Abdul Salam
Artificial Intelligence Dept., Faculty of Computers and Artificial Intelligence, Benha University, Egypt
Faculty of Computer Studies, Arab Open University, Cairo, Egypt

Mohamed Abdel-Fattah
Information System Dept., Faculty of Computers and Artificial Intelligence, Benha University, Egypt

Abdullah Abdel Moemen
Information System Dept., Faculty of Computers and Artificial Intelligence, Benha University, Egypt

ABSTRACT

Finding, locating, and resolving software defects takes a lot of time and effort on the part of software engineers. Humans are required to search and analyses data in traditional testing. Humans are prone to making incorrect assumptions, resulting in distorted results, which leads to defects being undetected. Machine learning enables systems to learn and use what they have learnt in the future, providing software testers with more accurate information. Several advanced machine learning approaches, such as deep learning, are capable of performing a variety of software engineering tasks, including code completion, defect prediction, bug localization, clone detection, code search, and learning API sequences. One of the most essential methods of examining software quality assurance is software testing. This procedure is time-consuming and costly, accounting for over half of the total cost of software development. Researchers are looking for using automated methods to reduce the cost and time of the test, in addition to the cost issue. A survey has been conducted with comparison between Machine Learning, and Data Mining algorithms.These algorithms are such as: Hill-Climbing Algorithm (HCA), Artificial Bee Colony Algorithm (ABC), Firefly Algorithm (FA), Particle Swarm Optimization (PSO), Artificial Bee Colony Algorithm (ABC), Genetic Algorithm (GA), Ant Colony Optimization (ACO), Artificial Neural Network (ANN), Support Vector Machine (SVM) and Hybrid Algorithms.

Keywords

Machine learning, artificial intelligence, data mining, Software Testing, Machine Learning, Testing Automation, Software Testing Tool.

1. INTRODUCTION

Testing conducted on a software product is called software testing. The main objective of testing or software testing is to find bugs in the software. Bug is an error or fault caused in the behavior of the software program or software application. Software testing can be done to check if the software

- Meets all the requirements mentioned in design phase
- Gives correct output for different inputs
- Will be able to complete the task within time limit or acceptable time.
- Will run in different environments.

Test cases are a set of conditions used by testers to determine whether or not the system under test operates properly. The creation of test cases aids in the discovery of application flaws or needs [1].

Any software application testing that is automated will go through a series of activities, processes, and tools in order to complete the test. The outcomes of these runs can be saved and recorded [3].

Software Testing can be majorly classified into two categories:

1. Black Box Testing is a software testing method in which the internal structure/ design/ implementation of the item being tested is not known to the tester.

2. White Box Testing is a software testing method in which the internal structure/ design/ implementation of the item being tested is known to the tester [4].

Test automation can be of 2 types:

1. Testing with code: uses pre-existing interfaces, libraries, classes, and modules to test with a large number of inputs and check and verify whether or not the results are right.
2. Testing with a Graphical User Interface (GUI): Interface events such as keystrokes and mouse clicks can be generated and used by the framework to identify changes and test whether the program's performance is right or not [5].

The benefits of automation testing:

1. Speed and Faster time to market
2. Wider test coverage
3. Consistency
4. Cost savings
5. Frequent and thorough testing

There are some instances where manual testing is better than automation testing, including (see Figure 1):

- New test cases that have not yet been executed manually.
- Test cases where the criteria are always changing.
- Test cases that are not routine [2].

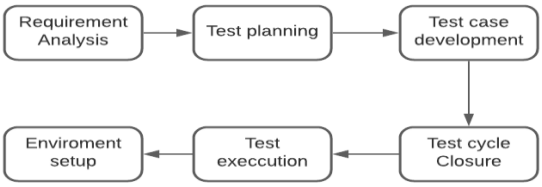

Fig 1: Software test cycle [2]

Once the application is developed based on the below parameters the product is verified and validated in testing environment such as:

- Reliability
- Portability
- Usability
- Flexibility
- Testability
- Efficiency

The main objective of this study is to build a techniques for pinpointing and correcting bugs in the software (see Figure 2), solve the issues in software testing automation, help software testers with more accurate knowledge, enhance the accuracy and minimize effort and time of software testing. These improvements can be achieved by:

1. Implement the data mining algorithms, machine learning techniques and AI methodology to obtain new model for automated software testing.
2. Working to choose the most efficient and appropriate learning method for automation of a target testing stage.
3. Evaluating the performance of the proposed work with the previous works to touch on the changes of data mining and machine learning techniques.

The idea improves the current test management life cycle by making it more productive, efficient, and a rapid and reliable self-decision-making system. The innovation uses supervised / semi-supervised learning to automate and save time on all processes. Algorithms and algorithms regulate quality such that any number of problems can be dealt with quickly. Almost 90% of the applications are planned and developed through this control process. The major challenges in reliability growth models are that every tester must include an operational profile before testing the product, which must specify the operations that a low number of failures defines a reliable product, and that fault assumptions in software reliability models must be independent. Because it is necessary to specify the test efforts, including the testing time and finish time. Software reliability provides engineers with a solution by anticipating application requirements from the beginning to the end.

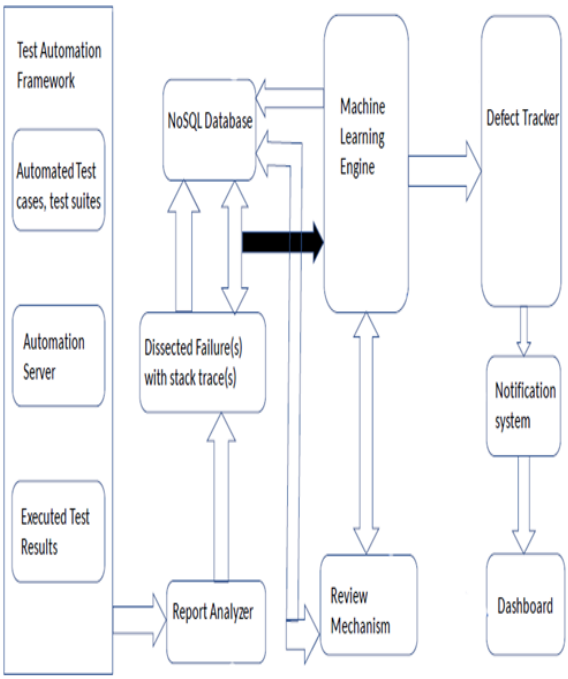

Fig 2: Illustrates the system for automated software testing based on ML [6]

The steps of automated software testing (see Figure 3):

1) Software test automation framework configured to collect automated test suite and test execution results.
2) A report parser to parse the test execution results generated by the software test automation framework and configured to identify the failures or exceptions with their respective stack trace.
3) A NoSQL database configured to hold historical defect, bug tickets with past failures or exceptions.
4) A ML engine to evaluate matching results of the NoSQL database and configured to predict type of the failure or exception.
5) A defect - tracking tool configured to create relevant bug tickets based on the type of failure or exception.
6) An automated notification system configured to notify the status of a bug ticket.
7) A dashboard to facilitate the access results, logs, failures, key performance indicators etc. in the form of histograms, pie graphs etc.
8) A manual feedback mechanism for adjusting the machine learning algorithm and NoSQL database table entries.

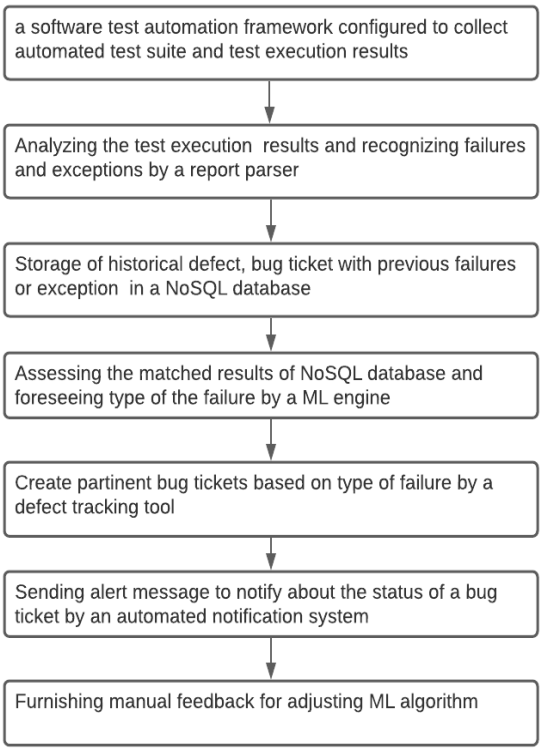

Fig 3: Illustrates a method for automated software testing based on ML, in accordance to one or more embodiment of the present invention [6]

Although applying ML to tackle software-testing problems is a relatively new and emerging research trend, many studies have been published in the past two decades. Different ML algorithms have been adapted and used to automate software testing, however, it is not clear how research in this area has evolved in terms of what has already been investigated. Despite the inherent value of examining the nature and scope of the literature in the area, few studies have attempted to provide a general overview of how ML algorithms have contributed to efforts to automate software-testing activities. For instance, proposed a frame- work that can be used to classify research at the intersection of ML and software testing. Nevertheless, their classification framework is not based on a systematic review of the literature, which to some extent undermines the scope and validity of such framework [7].

2. LITERATURE REVIEW

Mehdi Esnaashari, Amir Hossein Damia, they discussed that The main issue in the test data generation process is to determine the input data of the program in such a way that it meets the specified test criterion, in this study, a structural method was employed to automate the test data creation process while keeping the condition of covering all finite pathways in mind. The problem is transformed into a search problem in structural approaches, and meta-heuristic algorithms are utilized to solve it. This work proposes a memetic algorithm that employs reinforcement learning as a local search approach within a genetic algorithm [8].

The conclusion in this paper (see Figure 4) the success rate of MAAT in comparison to other existing algorithms. It is evident from this figure that the proposed algorithm has a success rate of 100%, while none of the other algorithms can reach more than 80% in this criterion. This is again due to the fact that mean fitness evaluations at MAAT is much lower than other algorithms.

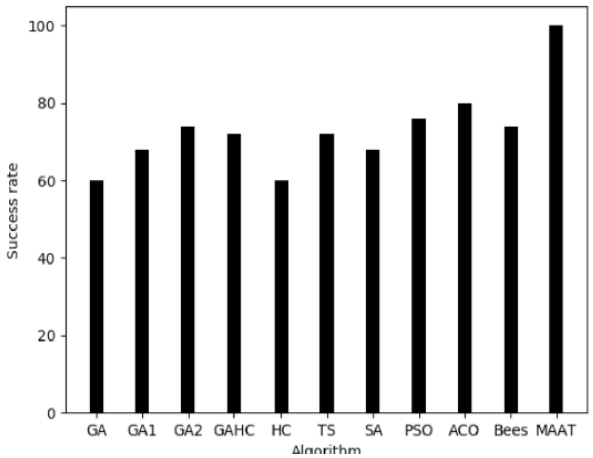

Fig 4: comparison success rate and algorithms for Experiment (Triangle program) [8]

Dr. Subarna Shakya, Dr. S. Smys, they suggested research study develops an optimized automatic software testing model using a hybrid model of differential evolution and ant colony optimization to increase software testing accuracy and reliability. To validate the proposed model's reliability, it is compared to traditional models such as artificial neural networks and particle swarm optimization [9, 21, and 22].

The experimentation of proposed hybrid model is performed in Mat Lab 14.1 installed on a 3.5 GHz i3 processor with 4GB of RAM. To increase the complexity of the experimentation process, we have used three different data sets with different attributes [23, 24].

The pseudocode for the proposed hybrid optimization model is summarized as follows.

Initialize ant population size,
Based on continues trail generate random variables
Compare the strength and its stability factor
Select optimal strategy policy
Check and change the next preferred location
The parameters are $\varphi_2, \varphi_3, \ldots \varphi_n$
for $h_l = Rand\ int(\varphi_n l)$
Select the random points $p^n, n = 1,2, \ldots . h_n$
for $n = 1\ to\ h_n$
computing the integer values $h_{best} = int(\varphi_4 h)$
if $h_{best} > 0$ *then*
for $i = 1\ to\ h_{best}$
move and create a suitable connection the set $Z_\sigma = z(x_\sigma^i, y_\sigma^j)$
If $Z_\sigma < Z_n$ *then replace the current position into new position based on the function value.*

Fig 5: Proposed Model (H-ACO) [9]

The accuracy comparison between ANN, PSO and H-ACO is depicted and it is observed that proposed model attains better accuracy compared to other models. The proposed model hybrid ACO model attains accuracy range of rate of 96.2%, but particle swarm optimization attains an average of 95.5% and artificial neural network obtains an accuracy of 92% which is 4% lesser than the proposed model.

The performance comparison in terms of sensitivity and specificity is depicted. It is observed from the figure, that the proposed model attains better sensitivity and specificity values than artificial neural network and particle swarm optimization model.

The proposed model (see Figure 5) attains better reliability compared to other model by identifying the bugs in datasets with an accuracy rate of 96.2%.

(see Figure 6) depicts a computation time analysis. All three models are tested on all three data sets. When compared to other models, the proposed model takes the least amount of time to compute. The performance of ANN lags due to its low memory capacity, while PSO takes more time due to its convergence parameters.

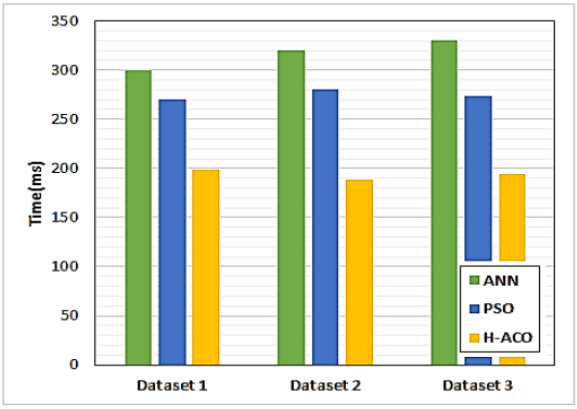

Fig. 6. Computation Time Comparison [9]

Mukesh Mann, Om Prakash Sangwan, and Pradeep Tomar, Smys, they work for the design and execution of test cases, this study proposes a regression framework based on a keyword-oriented data-driven methodology. The created framework's methodology is based on Test Language Processing (TLP), which is a complete approach to test case design and execution. Vtiger-Customer Relationship Management (CRM) version 5 is an open-source web application that is used to test the framework. In terms of test suite execution and optimization, the framework is compared to manual testing [10].

Based on our experiments it is concluded that (1) Test execution time using TLP based framework is significantly low and (2) a test suite optimization of 83.78% is achieved through the proposed TLP framework. For test suite execution and optimization in the shortest amount of time, a Regression Framework (based on TLP) is built. With further investigation, it was shown that the time value of manual testing varies with each subsequent iteration since manual testing is dependent on human efficiency, which is dependent on a variety of characteristics such as experience and domain expertise of the tester. However, it has been discovered that, in terms of time, subsequent cycles of iterations do not play a significant role in the established framework. MUT's testing time remains constant with each iteration. As a result, this framework aids in the speedy execution of testing [10].

SiwakornSrisakaokul, ZhengkaiWu, AngelloAstorga, OreoluwaAlebiosu, Tao Xie, they offer a method for testing supervised learning software, which is a common form of machine learning software, using multiple implementation testing. Our method, in particular, generates a test input's proxy oracle from the majority-voted output of many implementations of the same algorithm's test input (based on a pre-defined percentage threshold). Our method classifies as failing tests those test inputs whose outputs (generated by the Implementation under test) differ from the majority-voted outputs. We test our method against two well-known supervised learning algorithms: k-Nearest Neighbor (kNN) and Naive Bayes (NB) [11].

Conclusion is an approach of multiple implementation testing for supervised learning software. Our tests on two common machine learning algorithms, k-Nearest Neighbor (kNN) and Naive Bayes (NB), revealed that our majority-voted oracle, generated through multiple implementation testing, is a good surrogate for a test oracle. The oracle with the most votes has fewer false positives and can discover more true flaws than the oracle with the most votes. In particular, 19 kNN implementations detect 13 real errors and 1 potential fault, while 7 NB implementations detect 16 real faults. Among the three widely used open-source ML projects, our technique can detect 7 true problems and 1 potential defect [11].

Anna Trudova, Michal Dolezel, they purpose of this Systematic Literature Review (SLR) is to highlight the significance of artificial intelligence in software test automation by categorizing AI techniques and associated software testing tasks to which they might be used. The impact of AI on those activities was specifically investigated. To that purpose, the SLR was focused on research studies that reported on the use of AI approaches in software test automation [12].

According to the collected data, most commonly used AI techniques appears to be from the field of machine learning, specifically different types of neural networks: Artificial Neural Network, Recurrent Neural Network, Bayesian Network; Q-learning; L* etc. Bayesian Network and techniques from the Computer Vision field belong among the techniques that were used across more testing activities more frequently than others.

Akshat Sharma, Rishon Patani and Ashish Aggarwal, we work on describes a collection of approaches for automatically generating test data in software testing that employs a genetic algorithm. Researchers have proposed numerous ways for generating test data over the years, each with its own set of problems. We have provided numerous Genetic Algorithm (GA) based test techniques in this research, each with its own set of parameters for automating the generation of structural-oriented test data on the basis of internal programmer structure [13].

Need for Genetic Algorithms in Software Testing [25, 26 and 27]:

- Drawbacks of manual testing: [17, 18]
- Speed of operation is limited as it is carried out by humans.
- High investment in terms of cost, time.
- Limited availability of resources.
- Redundancy in test cases.
- Inefficient and inaccurate test checking.
- Pros of using genetic algorithms in software testing:
- Parallelism is a important characteristic of genetic testing [19, 20].
- Less likely to get stuck in extreme ends of a code during testing since it operates in a search space.

- With the same encoding, only fitness function needs to be changed according to the problem.

There are few basic methodology/Terminology that will be used while implementing genetic algorithm like:

- Individual – Possible solutions
- Population - Set of all individuals
- Search Space - All possible solutions to the specified problem
- Chromosome – Blueprint for an individual
- Trait - Possible aspect of an individual entity.
- Allele - Possible settings for a trait
- Locus - The position of a gene on the chromosome
- Genome - Collection of all chromosomes for an individual entity.

Algorithm:

- Start with randomly generated test cases from the population.
- Calculate the fitness f(x) of each pair of test cases (chromosome x) in the population.
- Repeat the following steps until a n child test cases have been generated.

The steps of algorithm:

a) Select a pair of parent test cases from the current population, with selection probability increasing as fitness increases. The selection is done "with replacement," which means that the same pair of test cases can be chosen as a parent many times. That is to say (Selection process is carried out).
b) Using the crossover probability Pc, cross the couple at a random location to create two kid cases or offspring. Form two test cases that are exact clones of their respective parent cases if no crossover occurs.
c) Mutate the two kid cases with Pm mutation probability and add the resulting pair of test cases to the new population. If n is odd, a new member of the population can be rejected at random. Substitute the new test cases for the existing ones.

Genetic Algorithm Implementation in C++, The outer loop, which will generate the remaining possible test cases, is the second phase for creating data. The loop will iterate until it satisfies the test findings for the supplied population values, accounting for the possibility of unrealistic test requirements such as branches and statement values. The algorithm will generate values for the crossover and mutation operators to use. The fitness function for individual values is next determined, followed by the population's individual average fitness functions. In the final step, the algorithm will assign the combined values of the test cases and find at least one individual desired fitness function values until enough test generations have been passed.

Even as the number of test cases grows, genetic algorithms become more efficient. Because data points in Random Testing Methods do not depend on time, it becomes inefficient as code becomes more sophisticated. As a result, Genetic Algorithms are being utilized to improve the efficiency and processing time of Software testing by providing us with an automatic test case generator. The evolutionary creation of test cases can be used, and it has been shown to be more efficient and cost effective than Random Testing.

Manju Khari, Anunay Sinha, Elena Verdu´, Ruben Gonza´ lez Crespo, they focuses on the performance evaluation of six metaheuristic algorithms namely: hill-climbing algorithm (HCA), particle swarm optimization (PSO), firefly algorithm (FA), cuckoo search algorithm (CS), bat algorithm (BA) and artificial bee colony algorithm (ABC) by using their standard implementation to optimize the path coverage and branch coverage produced by the test data [14, 28].

The results for hundred executions of each algorithm to generate the most optimized test suites for the five Java programs under test:

1) Triangle classification problem.
2) Greatest number problem.
3) Prime number identification problem.
4) Days in a month problem.
5) Binary search problem.

As a result, the results showed that ABC had the best balanced performance. When compared to the other algorithms, ABC generated the best results in the shortest amount of time. [97.8%, 98%, 50%, 97.92%, 80%] was the path coverage for ABC's five programs. These algorithms achieved the best path coverage when compared to other methods. Despite the fact that ABC was not the quickest algorithm, it produced the best results. Following BA, which was the fastest, ABC was found to be the second and third fastest algorithms, depending on the problem. Although BA was the fastest, its path coverage was less than ABC's [97%, 95%, 50%, 88.23%, 65%] was BA's path coverage. PSO was able to create optimal test suites, but it was slow. FA was determined to be the slowest of the algorithms, with results that were not comparable to the test suites of ABC and BA. CA had the worst results, and HCA, despite being quite fast, did not get good results.

According to the results, ABC achieved the best tradeoff between process and product metrics in the TSG problem. ABC, BA and PSO were the better optimal test suite generators, while CA, HCA and FA produced non-optimal test suites. On the other hand, BA, HCA and ABC were the faster algorithms with similar processing times for the process metrics. FA, PSO and CA were among the slower performing algorithms. Hence, ABC and BA present as suitable algorithms for TSG, while PSO can be improved in the future for the special case of TSG [29, 30].

Mohd. Mustaqeem, Mohd. Saqib, they discussed that the previous research conducted on the data without feature reduction lead to the curse of dimensionality. To solve the problem, we used a machine learning hybrid strategy that combined Principal component Analysis (PCA) and Support vector machines (SVM). To perform our research, we used PROMISE [15] (CM1: 344 observations, KC1: 2109 observations) data from NASA's directory. The dataset was divided into two parts: training (CM1: 240 observations, KC1: 1476 observations) and testing (CM1: 104 observations, KC1: 633 observations) [16].

The proposed model is the combination of the following two models PCA and SVM. The SVM is strong enough to classified defects and non-defective software observations. But, we have also included PCA to reduce the time complexity and robustness of the analysis. PROMISE data

repository datasets like KC1and MC1are full of various attributes which required high processing power. This paper also explains clearly how high dimensionality data goes inside the PCA (red color dotted rectangle part). Then, low dimension data are given to SVM to do classification (green color dotted rectangle part). F-measures, Recall, Accuracy, and Precisions are used to calculate test outcomes. On the KC1 dataset, we discovered that PC-SVM provided accuracy of 86.6 percent with 86.8% precision, 99.6% recall, and 92.8 percent F-measure. Similarly, the accuracy of the CM1 dataset-specific model was 95.2 percent, with 96.1 percent precision, 99 percent recall, and 97.5 percent F-measure.

3. CONCLUSION AND FUTURE WORK

In the first, this survey provides what is testing, automation testing, and types of software testing and benefits of automation testing. Also, the survey discussed some paper in automation software testing using machine learning and data mining algorithms. Also, this paper contains some techniques to solve the problem of test suite generation and test suite optimization, how to fetch and predict bugs and solve the issues in software testing automation, help software testers with more accurate knowledge, enhance the accuracy and minimize effort and time of software testing. The most algorithms used for automation testing are [PC-SVM, GA, KNN and ACO] (see Table 1), these techniques give the high result in accuracy and minimize the time of automation testing and effort. Some improvements on used models will be added by using metaheuristic optimization algorithms to find the best solution.

Some improvement on chooses the most efficient and appropriate learning method for automation and build a hybrid model of the pervious data mining, machine learning and swarm intelligence algorithms in software testing automation to solve the issue in generate test case or fetch bugs in software [31-40].

Table 1: The conclusion work of the survey

ID	Name	Year	Method	Result
1	Automation of software test data generation using genetic algorithm and reinforcement learning	2021	MAAT Algorithm	MAAT algorithm has a success rate of 100%, while none of the other algorithms can reach more than 80% in this criterion
2	Reliable Automated Software Testing Through Hybrid Optimization Algorithm	2020	Hybrid ACO Algorithm	The proposed model hybrid ACO model attains accuracy range of rate of 96.2%, but particle swarm optimization attains an average of 95.5% and artificial neural network obtains an accuracy of 92% which is 4% lesser than the proposed model
3	Automated Software Test Optimization using Language Processing	2019	TLP based framework	Based on our experiments it is concluded that (1) Test execution time using TLP based framework is significantly low and (2) a test suite optimization of 83.78% is achieved through the proposed TLP framework
4	Multiple-Implementation Testing of Supervised Learning Software	2018	k-Nearest Neighbor (kNN) and Naive Bayes (NB)	In particular, 19 kNN implementations detect 13 real errors and 1 potential fault, while 7 NB implementations detect 16 real faults. Among the three widely used open-source ML projects, our technique can detect 7 true problems and 1 potential defect
5	Artificial Intelligence in Software Test Automation: A Systematic Literature Review	2020	Systematic Literature Review (SLR)	Most commonly used AI techniques appears to be from the field of machine learning, specifically different types of neural networks: Artificial Neural Network, Recurrent Neural Network, Bayesian Network; Q-learning; L* etc. Bayesian Network and techniques from the Computer Vision field belong among the techniques that were used across more testing activities more frequently than others
6	Software Testing Using G Genetic Algorithm	2016	Genetic Algorithm (GA)	As a result, Genetic Algorithms are being utilized to improve the efficiency and processing time of Software testing by providing us with an automatic test case generator. The evolutionary creation of test cases can be used, and it has been shown to be more efficient and cost effective than Random Testing.
7	Performance analysis of six meta-heuristic algorithms over automated test suite generation for path coverage-based optimization	2019	hill-climbing algorithm (HCA), particle swarm optimization (PSO), firefly algorithm (FA), cuckoo search algorithm (CS), bat algorithm (BA)	ABC, BA and PSO were the better optimal test suite generators, while CA, HCA and FA produced non-optimal test suites. On the other hand, BA, HCA and ABC were the faster algorithms with similar processing times for the process metrics. FA, PSO and CA were among the slower performing algorithms. Hence, ABC and BA present as suitable algorithms for TSG, while PSO can be improved in the future for the special case of TSG

			and artificial bee colony algorithm (ABC)	
8	Principal component based support vector machine (PC-SVM)	2021	hybrid strategy that combined Principal component Analysis (PCA) and Support vector machines (SVM)	PC-SVM provided accuracy of 86.6 percent with 86.8% precision, 99.6% recall, and 92.8 percent F-measure. Similarly, the accuracy of the CM1 dataset-specific model was 95.2 percent, with 96.1 percent precision, 99 percent recall, and 97.5 percent F-measure

4. REFERENCES

[1] Mark Last, Menahem Friedman, Abraham Kandel, "The Data Mining Approach to Automated Software Testing", August 24-27, 2003, Washington, DC, USA.

[2] S.Sharmila, Dr Antony SelvadossThanamani, "Analytical Study of Data Mining Techniques for Software Testing", The International journal of analytical and experimental modal analysis. 6-december-2018.

[3] Ms.Karuturi Sneha, Mr. Malle Gowda M, "Research on Software Testing Techniques and Software Automation Testing Tools". International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS-2017).

[4] Jihyun Lee, Sungwon Kang, and Danhyung Lee, "A Survey on Software Testing Practices", All content following this page was uploaded by Sungwon Kang. 15 January 2015.

[5] Hussam Hourani, Ahmad Hammad, Mohammad Lafi, "The Impact of Artificial Intelligence on Software Testing", 2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT).

[6] Sumit Mahapatra and Subhankar Mishra, "Usage Of Machine Learning In Software Testing". July 11, 2020.

[7] Ashritha S and Dr. Padmashree T, "Machine Learning for Automation Software Testing Challenges, Use Cases Advantages & Disadvantages". International Journal of Innovative Science and Research Technology, September – 2020.

[8] Mehdi Esnaashari, Amir Hossein Damia, "Automation of software test data generation using genetic algorithm and reinforcement learning", Expert Systems With Applications 183 (2021) 115446 , 18 June 2021.

[9] Dr. Subarna Shakya, Dr. S. Smys, "Reliable Automated Software Testing Through Hybrid Optimization Algorithm". Journal of Ubiquitous Computing and Communication Technologies (UCCT) (2020).

[10] Mukesh Mann, Om Prakash Sangwan, and Pradeep Tomar, Smys, "Automated Software Test Optimization using Language Processing". The International Arab Journal of Information Technology, Vol. 16, No. 3, May 2019.

[11] SiwakornSrisakaokul, ZhengkaiWu, AngelloAstorga, OreoluwaAlebiosu, Tao Xie, "Multiple-Implementation Testing of Supervised Learning Software". 2018.

[12] Anna Trudova, Michal Dolezel, "Artificial Intelligence in Software Test Automation: A Systematic Literature Review", 2020.

[13] Akshat Sharma, Rishon Patani and Ashish Aggarwal, "SOFTWARE TESTING USING GENETIC ALGORITHMS", International Journal of Computer Science & Engineering Survey (IJCSES) Vol.7, No.2, April 2016.

[14] Manju Khari, Anunay Sinha, Elena Verdu´, Ruben Gonza´ lez Crespo, "Performance analysis of six meta-heuristic algorithms over automated test suite generation for path coverage-based optimization", Published online: 17 October 2019. Springer-Verlag GmbH Germany, part of Springer Nature 2019.

[15] Mohd. Mustaqeem, Mohd. Saqib, "Principal component based support vector machine (PC-SVM): a hybrid technique for software defect detection", part of Springer Nature 2021, Published online: 16 April 2021.

[16] Sayyad Shirabad, J., Menzies, T.J.: The {PROMISE} Repository of Software Engineering Databases (2005).

[17] Christoph C. Michael, Gary E. McGraw, Michael A. Schatz, and Curtis C. Walton, "Genetic Algorithms for Dynamic Test Data Generation," Proceedings of the 1997 International Conference on Automated Software Engineering (ASE'97) (formerly: KBSE) 0-8186-7961-1/97 © 1997 IEEE.

[18] Praveen Ranjan Srivastava et al, "Generation of test data using Meta heuristic approach" IEEE TENCON (19-21 NOV 2008), India available in IEEEXPLORE.

[19] Nashat Mansour, MiranSalame," Data Generation for Path Testing", Software Quality Journal, 12, 121–136, 2004,Kluwer Academic Publishers.

[20] FranciscaEmanuelle et. al., "Using Genetic algorithms for test plans for functional testing", 44th ACM SE proceeding, 2006, pp. 140 - 145.

[21] Ajmer Singh, Rajesh Bhatia, Anita Singhrova (2018). Taxonomy of machine learning algorithms in software fault prediction using object-oriented metrics. Procedia Computer Science. 132. 993-1001.

[22] Alireza Haghighatkhah, Ahmad Banijamali, Olli-PekkaPakanen, Markku Oivo, PasiKuvaja (2017).

Automotive software engineering: A systematic mapping study. Journal of Systems and Software. 128, 25-55.

[23] Amir Elmishali, Roni Stern, Meir Kalech (2018). An Artificial Intelligence paradigm for troubleshooting software bugs. Engineering Applications of Artificial Intelligence. 69, 147-156

[24] Fuqun Huang, Bin Liu (2017). Software defect prevention based on human error theories. Chinese Journal of Aeronautics. 30(3):1054-1070.

[25] Mark Last, Shay Eyal1, and Abraham Kandel, "Effective Black-Box Testing with Genetic Algorithms," IBM conference.

[26] Wegener, J., Baresel, A., and Sthamer, H, "Suitability of Evolutionary Algorithms for Evolutionary Testing," In Proceedings of the 26th Annual International Computer Software and Applications Conference, Oxford, England, August 26-29, 2002.

[27] Mark Last et. al., "Effective black-box testing with genetic algorithms", Lecture notes in computer science, Springer, 2006, pp. 134 -148.

[28] Kumudha, P., Venkatesan, R.: "Cost-sensitive radial basis function neural network classifier for software defect prediction". Sci.World 2016, 2401496 (2016J.)

[29] Hudaib, A., Zaghoul, F.A.L., Widian, J.A.L.: "Investigation of software defects prediction based on classifiers (NB, SVM, KNN and decision tree)". J. Am. Sci. 9(12), 381–386 (2013).

[30] Naughton, J.: "The evolution of the Internet: from military experiment to general purpose technology". J. Cyber Policy 1(1), 5–28 (2016).

[31] O. Hegazy, O.S. Soliman, and M. Abdul Salam, "A Machine Learning Model for Stock Market Prediction", International Journal of Computer Science and Telecommunications, Vol. (4), Issue (12),pp. 17-23, December 2013.

[32] O. Hegazy, O.S. Soliman, and M. Abdul Salam, "LSSVM-ABC Algorithm for Stock Price Prediction", International Journal of Computer Trends and Technology (IJCTT), Vol. (7), Issue (2),pp. 81-92, Jan 2014.

[33] O. Hegazy, O.S. Soliman, and M. Abdul Salam, "Optimizing LS-SVM using Modified Cuckoo Search algorithm (MCS) for Stock Price Prediction", International Journal of Advanced Research in Computer Science and Management Studies, Vol. (3), Issue (2),pp. 204-224, February 2015.

[34] O. Hegazy, O.S. Soliman, and M. Abdul Salam, "Comparative Study between FPA, BA, MCS, ABC, and PSO Algorithms in Training and Optimizing of LS-SVM for Stock Market Prediction", International Journal of Advanced Computer Research Vol.(5), Issue (18),pp.35-45, March-2015.

[35] O. Hegazy, O.S. Soliman, and M. Abdul Salam, "FPA-ELM Model for Stock Market Prediction", International Journal of Advanced Research in Computer Science and Software Engineering, Vol.(5), Issue (2),pp.1050-1063, February 2015.

[36] R. Salem, M. Abdul Salam, H. Abdelkader and A. Awad Mohamed, "An Artificial Bee Colony Algorithm for Data Replication Optimization in Cloud Environments," in IEEE Access, vol. 8, pp. 51841-51852, 2020, doi: 10.1109/ACCESS.2019.2957436.

[37] M. A. Salam, A.T. Azar, R. Hussien, R. (2022). Swarm-Based Extreme Learning Machine Models for Global Optimization. CMC-COMPUTERS MATERIALS & CONTINUA, 70(3), 6339-6363.

[38] M. Abdul Salam, S. Taha, M. Ramadan. COVID-19 detection using federated machine learning. Plos one. 2021 Jun 8;16(6):e0252573.

[39] O. Hegazy, O.S. Soliman, and M. Abdul Salam, A Hybrid BA-LS-SVM Model and Financial Technical Indicators for Weekly Stock Price and Trend Prediction. International Journal. 2014 Apr;4(4).

[40] M. Abdelsalam, H. Ahmed, W.F. Abdulwahed, (2014). Evaluation of Differential Evolution and Particle Swarm Optimization Algorithms at Training of Neural Network for prediction. IJCI. International Journal of Computers and Information, 3(1), 2-14.