

Novel preemptive intelligent artificial intelligence-model for detecting inconsistency during software testing

Sangeetha Govinda¹, B. G. Prasanthi², Agnes Nalini Vincent³

¹Department of Computer Science, Christ University, Bangalore Central Campus, Bangalore, India

²Department of Computer Science and Applications, St. Josephs University, Bangalore, India.

³Faculty of Information Technology, AMITY Institute of Higher Education, Quatre Bornes, Mauritius

Article Info

Article history:

Received Apr 10, 2024

Revised Feb 26, 2025

Accepted Mar 15, 2025

Keywords:

Artificial intelligence

Automation

Error

Inconsistency

Software testing

ABSTRACT

The contribution of artificial intelligence (AI)-based modelling is highly significant in automating the software testing process; thereby enhancing the cost, resources, and productivity while performing testing. Review of existing AI-models towards software testing showcases yet an open-scope for further improvement as yet the conventional AI-model suffers from various challenges especially in perspective of test case generation. Therefore, the proposed scheme presents a novel preemptive intelligent computational framework that harnesses a unique ensembled AI-model for generating and executing highly precise and optimized test-cases resulting in an outcome of adversary or inconsistencies associated with test cases. The ensembled AI-model uses both unsupervised and supervised learning approaches on publicly available outlier dataset. The benchmarked outcome exhibits supervised learning-based AI-model to offer 21% of reduced error and 1.6% of reduced processing time in contrast to unsupervised scheme while performing software testing.

This is an open access article under the [CC BY-SA](#) license.



Corresponding Author:

B. G. Prasanthi

Department of Computer Science and Applications, St. Josephs University

36, Langford Rd, Langford Gardens, Bengaluru, Karnataka 560027, India

Email: prasanthi.b.g@sju.edu.in

1. INTRODUCTION

The mechanism of software testing plays a crucial and integral role in software development that consists of a critical analysis of overall quality to ensure that it meets demanded specification [1]. The primary agenda of different types of software testing is to perform validation, verification, identification of defects, and enhancing the quality [2]. However, there are some essential challenges associated with software testing [3]. Some of the notable challenges are complexity of software, changing requirements, time and resource, lack of test data, dependency management, automation challenges, non-deterministic behavior, platform and environment diversity, and performance and scalability testing [4]–[7]. At present, it is noted that artificial intelligence (AI) has a significant contribution towards incorporating automation in software testing thereby leveraging the efficiency of testing procedure [8]–[10]. There are various studies to showcase that automated generation of test-cases can be done by AI algorithms for given code, specification, and software requirements [11]. Such form of generated test-cases can be helpful for covering varied forms of scenarios and competitive cases in order to accomplish a broader spectrum of testing. AI also assists in offering test prioritization that can effectively analyze adversaries and all risk elements connected with different components of software and accordingly prioritizes the test cases [12]. Such features of test prioritization can be used for more emphasis towards rigorous testing on potentially extensive defects and hence resources of testing are highly optimized unlike conventional testing. AI-powered testing tools can

automate the execution of test cases, reducing the need for manual intervention and speeding up the testing process. This includes both functional and non-functional testing, such as regression testing, performance testing, and security testing [13]. AI algorithms can analyze historical data from previous testing cycles to predict potential defects or areas of the software that are more likely to contain bugs. This helps testers focus their efforts on high-risk areas and allocate resources more effectively [14]. AI can act as dynamic test oracles by learning the expected behavior of the software through training on historical data or user interactions. It can then compare the actual behavior of the software during testing with the expected behavior and identify deviations or anomalies [15]. AI techniques, such as machine learning and anomaly detection algorithms, can identify unexpected patterns or deviations from normal behavior in the software under test. This can help detect subtle bugs or security vulnerabilities that may not be apparent through traditional testing methods [16]. Natural language processing (NLP) can be used to analyze and understand natural language requirements, user stories, and documentation. AI-powered tools can extract testable scenarios and generate test cases directly from this textual information, improving test coverage and accuracy [17].

However, there are various challenges of existing system of AI in software testing as following: i) the primary challenge is related to overfitting by generating test cases that are only relevant to the specific training scenarios and may not generalize well to new, unseen situations; ii) it was also noted that AI-generated test cases may not cover all possible scenarios or edge cases, leading to gaps in test coverage. Balancing between generating enough diverse test cases and avoiding redundant or irrelevant ones is a challenge; iii) AI algorithms may struggle to understand and model the intricacies of complex software systems accurately. This can result in the generation of test cases that overlook critical interactions or dependencies within the system, leading to incomplete testing; iv) systems that exhibit dynamic behavior or have frequent changes pose challenges for AI-based test case generation. AI models may struggle to adapt quickly to changes in the software or the testing environment, leading to outdated or ineffective test cases; and v) AI may struggle to accurately define the expected outcomes for test cases, especially in complex systems or when requirements are ambiguous. Without clear oracles, it's challenging to assess whether the system behavior is correct, leading to difficulties in validating the generated test cases.

The related work carried out towards AI implementation in software testing are as follows: Robisco and Martínez [18] have developed a risk evaluation scheme using machine learning and NLP in order to detect the defaulters. Evaluation of the executable files has been carried out by Arakelyan *et al.* [19] where convolution network using graphs has been implemented towards semantic analysis of the vulnerability of data. Adoption of AI has also been witnessed in investigating the android-based application where a generation of stateful event has been discussed in work of Yerima *et al.* [20] using machine learning. Hai *et al.* [21] have used deep learning approach for tracking software errors present in cloud application for detecting software bugs using multilayered perceptrons. Further adoption of multilayered perceptron was also discussed by Cui *et al.* [22] where the idea is to generate suitable test cases using historical data. Jammalamadaka and Parveen [23] have used deep belief network in order to evaluate coverage of software testing with multiple datasets. Laranjeiro *et al.* [24] have developed an intelligent model towards detecting sub-optimal data quality associated with web services and applications. Martin [25] have presented a comprehensive analysis of multiple machine learning models to find the effectiveness towards predictive software testing. Adoption of unsupervised machine learning is witnessed in work presented by Sebastian *et al.* [26] with a target to minimize the test cases thereby minimizing the cost and time involved in processing test cases. Moghadam *et al.* [27] have used bio-inspired approaches in AI and machine learning in order to generate suitable test cases a use case of lane discipline in road transport. The work carried out by Ahmed *et al.* [28] have presented a unique scheme that can prioritize test cases while performing regression-based assessment. It was also noted that feature extraction plays one of the critical roles while applying machine learning in software testing. Study in such direction was carried out by Chen *et al.* [29] where the authors contribute towards potential feature selection that is necessary for predictive classification. The core research gap is that existing approaches adopts sophisticated AI scheme where the significant loopholes in evaluating software design quality is often unnoticed for their overfitting. Further, there is also a gap towards evolving out any low-cost involved algorithmic approach.

The prime contribution of the proposed study is to develop a novel intelligent computational framework to determine the inconsistencies within test ecosystem. The value-added contribution of the study are: i) to design a preemptive soft-computing model where AI has been implemented to determine adversary-based outcomes with false positives, ii) to develop an involuntary generation and execution of the optimal test cases with higher degree of reliability mapping with practical world dataset, iii) the prime agenda of this model is to generate a negative outcome by processing varied test cases in order to identify all possible forms of inconsistencies present within the given software design, iv) the proposed AI-model is designed considering revised version of both unsupervised and supervised learning scheme that generates a computed

predictive verdict towards trustworthy testcases, and v) a comprehensive benchmarking is carried out to investigate the error and algorithm processing time using unconventional methods of learning in AI-model.

2. METHOD

The prime aim of the proposed study is to design a computational framework that contributes towards generation of intelligent and appropriate test cases in order to assess the sustainability of the software prototype using AI. The architecture adopted for proposed study is shown in Figure 1. The novelty of proposed study is that it emphasizes on adverse testing outcome and not on normal testing outcome in order to figure out presence of abnormalities in analysis of data. The core idea of this model shown in Figure 1 is towards harnessing the potential of AI for authenticating the genuineness of adverse test outcome, determine potential crashes and failures within the assessment environment, and analyzed the outcomes obtained from the testing of the software. The proposed study also contributes towards developing an automated mechanism that is capable of sophisticated characteristic of software followed by correctly identifying the inconsistencies associated with software design.

The primary operation carried out by the proposed study model is to perform collection of data from the assessment eco-system while performing the evaluation of software robustness. In order to make the assessment scenario applicable for practical world, the study considers hardware in the loop environment for assessing the experimental prototype along with hardware integrated with software design. The model is analyzed using real-world sensory data of a mobile objects with an apriori rate of sampling. The dataset associated with training operation consists of test-cases of normal type and abnormal type. Further set of preprocessing operation of the sensory data is carried out followed by the training operation using AI model. The generated analytical model is used for determining the abnormalities localized within the considered environment that further generates the adverse test cases. The generated test outcome is further reported back to the testing professional in order to determine the reliability of adverse test resultant.

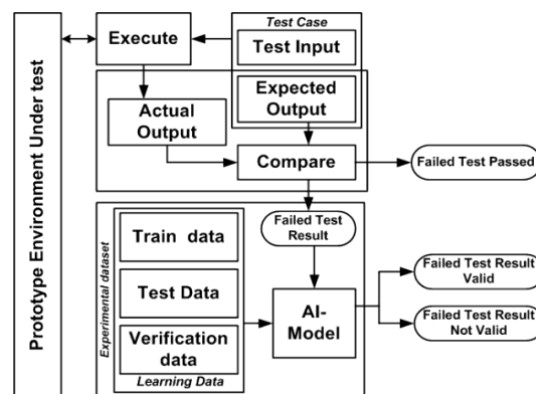


Figure 1. Proposed architecture

2.1. Preprocessing operation

This operation is carried out for the data that is aggregated in the process of evaluation with the test-cases that involves all the essential information associated with the test environment. It is to be noted that test outcome is basically the resultant of the execution of involuntary case of testing and this could be either negative or positive, depending on its nature of formation. When a negative outcome is generated owing to the assertion of even one condition, the scheme considers its test to be negative. The complete information associated with the behaviour of the test environment can be represented in the form of trace log objects that consists of all information associated with the dynamic changes of an environmental objects. The core idea of this part of implementation of preprocessing operation is to opt for exclusive and unique feature extracted from the test cases. As the complete outcomes of the testing can be stored in easily accessible client application interface, the proposed scheme can now perform simpler and faster extraction of features from this interface. The proposed scheme categorizes the data obtained from the trace log object into multiple classes which are mainly related to the environmental attributes and finally retains all the information in plaintext files for easier access to information. The process of data cleaning is carried out by eliminating the columns consisting of unnecessary data while any errors associated with formatting is repaired. The scheme then carries out a data parsing for investigating the entries of trace log objects followed by extracting the potentially significant information from the test outcome. A specific form of patterns on the basis of

environmental attributes are formed for the structured data obtained from trace log objects. The proposed scheme also extracts the information of the time involved for particular states of a dynamic system during the process of feature engineering. Further, label coding approach is used for acquiring numerical scores from the categorical attributes followed by performing stage of data transformation, where normalization is carried out for the recently acquired numerical attributes. 70-30 proportion rule is used for classifying the training and testing data. While applying AI-based approach, it is essential to offer more importance towards formation of the balanced data while carrying out the training that has a positive effect on performance. The next part discusses about the proposed AI-model.

2.2. Artificial intelligence-model

The proposed study towards AI-model is carried out using python owing to its robust and flexible supportability towards data science using its rich set of operators and libraries. A novel function is constructed using Python that is capable of determining the presence of any form of abnormalities and inconsistencies present within the software code design. Such form of function can be also used for both upcoming as well as current software project development in order to identify issues in evaluation data. The proposed study model is analyzed using integrated combination of multiple supervised and unsupervised machine learning algorithms as its AI-model. Following are the briefings of the learning approaches used in proposed study:

- Supervised learning: i) SL₁: the first supervised method is a combination of transformer-based attribute tokenizer and revised residual network [30], ii) SL₂: the next technique is a combination of multilayer perceptron with support vector machine that is meant for determining the defects [31], iii) SL₃: the third learning scheme is based on bundling of features integrated with sampling using gradient methods [32]. The sampling is carried out by eliminating all instances characterized by mini scale values of gradient while the remnant instances are utilized to evaluate the gain in information. The mechanism of feature bundling contributes towards feature reduction, and iv) SL₄: the fourth supervised approach implemented in proposed scheme is a unique method of executing gradient boosting [33] where permutation-based scheme is used deploying sequential boosting. This scheme can analyze features with categorical values.
- Unsupervised learning: i) USL₁: the assessment is carried out considering random forest in order to select optimal feature randomly for determination of inconsistencies in outcome. The approach performs allocation of abnormality value of observation based on temporal attribute for faster operation over larger and heterogeneous dataset [34], ii) USL₂: generative adversarial network has been implemented for similar reason towards reducing the reconstruction error [35], iii) USL₃: autoencoder algorithm has been further used for minimizing the reconstruction error thereby facilitating generation of extensive logical representation of an outcome [36], iv) USL₄: ensemble based k-nearest neighbour algorithm has been implemented for accomplishing faster detection of inconsistencies in adverse outcomes [37], v) USL₅: the proposed system also uses revised k-nearest neighbour algorithm where Hilbert curve is used for computing the weight targeting towards better convergence rate and minimized time complexity [38], vi) USL₆: the next approach used is based on dimensional reduction in order to minimize the computational burden as well as explore the latent patterns in data distribution [39], and vii) USL₇: the final unsupervised approach used in proposed scheme contributes towards presenting a non-parametric method of evaluating the possible input data distribution [40]. The end probability associated with all the data points are evaluated with this distribution. The next section presents discussion of result being accomplished.

3. RESULTS

The scripting of the proposed study is carried out in python environment considering publicly available open-access dataset [41] with a target to investigate the effectiveness of proposed AI model in order to detect the adverse outcome. The dataset consists of information related to proportion of inconsistencies, features, and size while the complete dataset is labelled which makes the task easier for analyzing both unsupervised and supervised learning approaches. The complete evaluation of the outcome is carried out considering two standard performance metric of mean squared error (MSE) and processing time of an algorithm. For an effective analysis, the study considering evaluation performance for MSE into two folds viz: i) evaluation of MSE for train images (MSE₁) and ii) evaluation of MSE for test images (MSE₂). The benchmarked numerical outcome is tabulated in Tables 1 and 2.

During the process of investigating the effectiveness of varied form of AI-models pertaining to determination of inconsistencies on both training dataset as well as testing dataset, the study outcome exhibited in above tabulated values shows some interesting outcomes over MSE and processing time score. From Table 1, the diversified outcome shows the effectiveness of some of the particular algorithms in AI-model on the basis of the data being used as an input. It was found that USL₄ using revised nearest neighboring is found with most effective outcome from the perspective of highly reduced MSE score for both

training (MSE=0.02) and testing data (MSE=0.11). Nearly similar performance is also observed for USL₃ using autoencoder (MSE_1=0.02 and MSE_2=0.17), USL₇ using non-parametric approach towards estimating inconsistencies (MSE_1=0.02 and MSE_2=0.16), USL₆ using dimensional reduction approach (MSE_1=0.024 and MSE_2=0.16).

From the perspective of supervised learning-based AI-models exhibited in Table 2, it is noted that optimal performance is shown by SL₁ approach that deploys transformer-based approach with residual network (MSE_1=0 and MSE_2=0.03). The second-best performance is exhibited by SL₃ (MSE_1=0 and MSE_2=0.05) and SL₄ (MSE_1=0 and MSE_2=0.05) using feature bundling with gradient sampling and gradient boosting respectively. However, a different form of performance trend is exhibited by SL₂ approach that uses multi-layered perceptron with support vector machine. Its uniqueness can be justified from reduced MSE_1=0.001 and MSE_2=0.01 during the testing which is reduced value in contrast to SL₁, SL₃, and SL₄ approaches in evaluation.

Table 1. Numerical outcome for unsupervised learning method

Unsupervised learning	MSE 1	MSE 2	Processing time
USL ₁	0.02	0.18	0.307
USL ₂	0.05	0.13	0.399
USL ₃	0.02	0.17	0.128
USL ₄	0.02	0.11	0.256
USL ₅	0.12	0.82	0.271
USL ₆	0.024	0.16	0.602
USL ₇	0.02	0.16	0.472

Table 2. Numerical outcome for supervised learning method

Supervised learning	MSE 1	MSE 2	Processing time
SL ₁	0	0.03	0.277
SL ₂	0.001	0.01	0.493
SL ₃	0	0.05	0.298
SL ₄	0	0.05	0.257

A closer look into the key findings of outcomes eventually shows that there are certain AI-models that exhibits much better performance while subjected to training dataset but was witnessed with over-fitting problems too when analyzed with testing data (e.g., SL₃ and SL₄). The reason for performance degradation in certain AI-models is due to the challenges encountered by these models during the process of generalizing untrained data. The evaluated score of MSE also interprets the possible accuracy in abnormality detection while performing predictive analysis. Further, a closer look into SL₂ algorithm shows that it offers better consistent performance of reduced error rate in both training dataset and testing dataset that is a direction representation of robustness associated with this particular AI-model. The trend of consistent lower MSE scores is also a direct indication of higher accuracy while performing predictive operation in proposed AI-model. Another intrinsic observation is that SL₂ model actually works exceptionally reliably well when used with only support vector machine; as inclusion of multi-layered perceptron was noted with reduced MSE performance for testing data and higher MSE score in training data. This is also an indication of higher sensitivity of multi-layered perceptron approach as a standalone towards training data. However, it is still recommended to integrate and use both multilayered perceptron and support vector machine together to reach the optimal state of outcome in proposed AI-model.

Figures 2 and 3 showcases the standalone outcome of both unsupervised and supervised learning model with respect to MSE respectively. As the number of learning approaches used in unsupervised are slightly more in contrast to number of supervised approaches, hence, the proposed scheme extracts the mean value of MSE_1, MSE_2, and processing time from Tables 1 and 2 for better inference. It shows that mean value of MSE_1 for unsupervised learning approaches is 0.03914 while mean value of MSE_2 is 0.24714, while the processing time mean score is found to be 0.34785s. Similar observation is carried out for mean value MSE_1 for supervised learning approach that is found to be 0.00025 while mean value of MSE_2 for supervised scheme is found to be 0.035. The mean value of processing time for supervised learning approach is found to be 0.33125 s. The outcome towards comparison showcases that supervised learning approaches offers better performance in contrast to unsupervised learning approaches. The mean score of supervised learning approaches for training (MSE_1) is found to be approximately 38% reduced compared to unsupervised learning. The key findings shows that the mean score of supervised learning approach for testing (MSE_2) is found to be approximately 21% reduced compared to that of supervised learning approach. The processing time for supervised learning is slightly increased compared to unsupervised learning approach by 1.6%, which is absolutely less significant and doesn't have any potential impact on computational effort used.

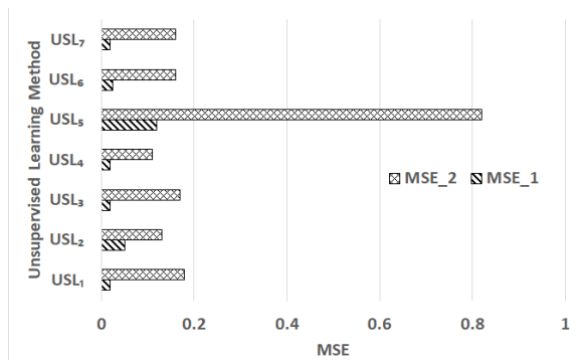


Figure 2. Benchmarked outcome of MSE for unsupervised learning

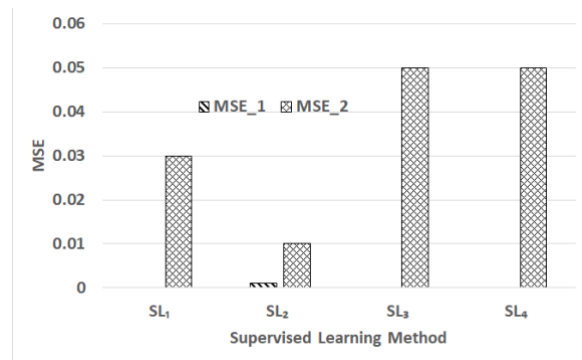


Figure 3. Benchmarked outcome of MSE for supervised learning

Figures 4 and 5 showcases the benchmarked outcome of processing time for unsupervised and supervised learning approaches to find that there is no significant difference between the mean score of processing time between both the AI approaches. However, from the perspective of granular study towards individual outcomes of each AI models, it is noted that unsupervised approach USL₃ using autoencoder offers the most reduced processing time ($t=0.128$ s). The performance of processing time for supervised approach SL₄ ($t=0.257$ s) and unsupervised approach USL₄ using revised ensembled nearest neighboring approach ($t=0.256$ s) are nearly similar. Such near similar processing time is also witnessed by supervised approach of SL₁ that uses residual network with transformer scheme ($t=0.277$ s) and unsupervised approach of USL₅ that uses revised k-nearest neighboring approach ($t=0.271$ s). Finally, it is noted that higher consumption of algorithmic processing time is exhibited by one supervised scheme of SL₂ using multilayered perceptron with support vector machine ($t=0.493$ s) and two unsupervised schemes of USL₆ using dimensional reduction ($t=0.602$ s) and USL₇ using non-parametric probability estimation method ($t=0.472$ s).

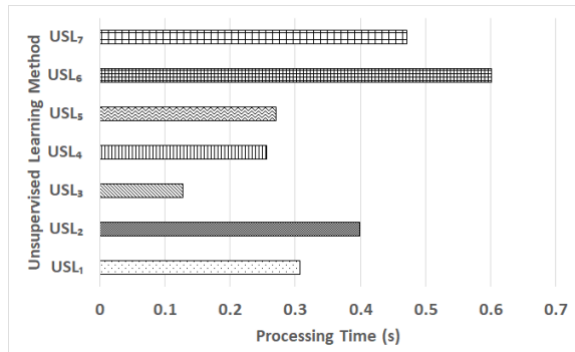


Figure 4. Benchmarked outcome of processing time for unsupervised learning

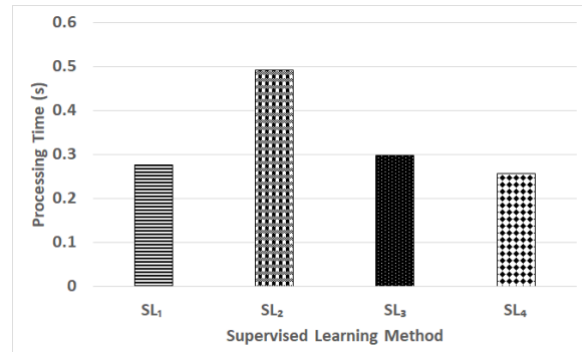


Figure 5. Benchmarked outcome of processing time for supervised learning

4. CONCLUSION

The prime agenda of this manuscript is to present a novel and intelligent computational approach that can harness the potential of AI in the process of validating the outcome during software design testing. At present, such forms of decision making of determining the adversary outcomes is done by expert system, while the proposed scheme hypothesizes that such decision making can be autonomously done by AI-models. The proposed study has investigation both unsupervised and supervised AI-model in order to accomplish much granularity in its outcome. Unlike the myth borned by industrial AI-based application testing that unsupervised is a better choice is proven otherwise in proposed investigation. The sub-optimal performance of unsupervised learning approach is witnessed in proposed investigation that can be justified on the ground of extensive processing time and higher error rates. The proposed study contributes towards incorporating following novel characteristics as follows: i) a novel evaluation platform is presented that is capable of determining the false positives and inconsistencies while performing testing of software design, ii) the proposed scheme presents a simplified mechanism of generating test cases considering experimental prototyping using publicly available

dataset of outliers, iii) the model is capable of executing varied form of test cases along with constructing new test cases in order to generate adversary outcome with negative results that as further recorded as trace log objects, iv) the model preprocesses the data followed by applying ensembled AI-models in order to perform detection of inconsistencies that are further analyzed to generate a verdict to assess their reliability, and v) the study findings showcases that supervised scheme of SL_2 using multilayered perceptron and support vector machine are higher recommended owing to reduced error score and optimal processing time involvement. The limitation of the proposed study is that it is not assessed with respect to all the use-cases where the performance can be improved using semi-supervised approaches. The future work will be carried out using a novel form of semi-supervised algorithm in order to investigate this perspective. Further, the model can be subjected to improvement considering more number of test-cases with temporal features.

ACKNOWLEDGMENTS

We would like to express sincere gratitude to Christ University and St.Joseph's university for providing the resources and support necessary for the completion of this research. The authors also acknowledge the contributions of Lab team and colleagues for their assistance with the research work.

FUNDING INFORMATION

Authors state no funding involved.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Sangeetha Govinda	✓	✓	✓	✓	✓	✓		✓	✓	✓			✓	
B. G. Prasanthi		✓	✓	✓		✓		✓	✓	✓	✓	✓		
Agnes Nalini Vincent	✓		✓	✓			✓			✓	✓		✓	✓

C : **C**onceptualization

M : **M**ethodology

So : **S**oftware

Va : **V**alidation

Fo : **F**ormal analysis

I : **I**nterpretation

R : **R**esources

D : **D**ata Curation

O : **O**riginal Draft

E : **E**xperimentation

Vi : **V**isualization

Su : **S**upervision

P : **P**roject administration

Fu : **F**unding acquisition

CONFLICT OF INTEREST STATEMENT

Authors state no conflict of interest.

DATA AVAILABILITY

The data that support the findings of this study are available from the corresponding author upon reasonable request.




REFERENCES

- [1] K. Salako and X. Zhao, "The unnecessary of assuming statistically independent tests in Bayesian software reliability assessments," *IEEE Transactions on Software Engineering*, vol. 49, no. 4, pp. 2829–2838, 2023, doi: 10.1109/TSE.2022.3233802.
- [2] M. Taromirad and P. Runeson, "A literature survey of assertions in software testing," *Engineering of Computer-Based Systems*, pp. 75–96, 2024, doi: 10.1007/978-3-031-49252-5_8.
- [3] S. Stradowski and L. Madeyski, "Exploring the challenges in software testing of the 5G system at Nokia: a survey," *Information and Software Technology*, vol. 153, 2023, doi: 10.1016/j.infsof.2022.107067.
- [4] O. Parry, G. M. Kapfhammer, M. Hilton, and P. McMinn, "A survey of Flaky tests," *ACM Transactions on Software Engineering and Methodology*, vol. 31, no. 1, 2021, doi: 10.1145/3476105.
- [5] S. Martínez-Fernández *et al.*, "Software engineering for AI-based systems: a survey," *ACM Transactions on Software Engineering and Methodology*, vol. 31, no. 2, 2022, doi: 10.1145/3487043.
- [6] V. Garousi, M. Felderer, M. Kuhrmann, K. Herkiloğlu, and S. Eldh, "Exploring the industry's challenges in software testing: an empirical study," *Journal of Software: Evolution and Process*, vol. 32, no. 8, 2020, doi: 10.1002/smr.2251.
- [7] T. Fulcini, R. Coppola, L. Ardito, and M. Torchiano, "A review on tools, mechanics, benefits, and challenges of gamified software testing," *ACM Computing Surveys*, vol. 55, no. 14 s, 2023, doi: 10.1145/3582273.
- [8] D. Amalfitano, S. Faralli, J. C. R. Hauck, S. Matalonga, and D. Distanto, "Artificial intelligence applied to software testing: a tertiary study," *ACM Computing Surveys*, vol. 56, no. 3, 2023, doi: 10.1145/3616372.




- [9] M. Boukhilif, M. Hanine, and N. Kharmoum, "A decade of intelligent software testing research: a bibliometric analysis," *Electronics*, vol. 12, no. 9, 2023, doi: 10.3390/electronics12092109.
- [10] F. A. Batareseh, R. Mohod, A. Kumar, and J. Bui, "The application of artificial intelligence in software engineering: a review challenging conventional wisdom," *Data Democracy: At the Nexus of Artificial Intelligence, Software Development, and Knowledge Engineering*, pp. 179–232, 2020, doi: 10.1016/B978-0-12-818366-3.00010-1.
- [11] A. Fontes and G. Gay, "The integration of machine learning into automated test generation: a systematic mapping study," *Software Testing Verification and Reliability*, vol. 33, no. 4, 2023, doi: 10.1002/stvr.1845.
- [12] J. A. P. Lima and S. R. Vergilio, "Test case prioritization in continuous integration environments: a systematic mapping study," *Information and Software Technology*, vol. 121, 2020, doi: 10.1016/j.infsof.2020.106268.
- [13] J. J. Li, A. Ulrich, X. Bai, and A. Bertolino, "Advances in test automation for software with special focus on artificial intelligence and machine learning," *Software Quality Journal*, vol. 28, no. 1, pp. 245–248, 2020, doi: 10.1007/s11219-019-09472-3.
- [14] J. Pachouly, S. Ahirrao, K. Kotecha, G. Selvachandran, and A. Abraham, "A systematic literature review on software defect prediction using artificial intelligence: Datasets, data validation methods, approaches, and tools," *Engineering Applications of Artificial Intelligence*, vol. 111, 2022, doi: 10.1016/j.engappai.2022.104773.
- [15] A. Gartzandia *et al.*, "Machine learning-based test oracles for performance testing of cyber-physical systems: an industrial case study on elevators dispatching algorithms," *Journal of Software: Evolution and Process*, vol. 34, no. 11, 2022, doi: 10.1002/smr.2465.
- [16] V. Rousopoulou *et al.*, "Cognitive analytics platform with AI solutions for anomaly detection," *Computers in Industry*, vol. 134, 2022, doi: 10.1016/j.compind.2021.103555.
- [17] V. Garousi, S. Bauer, and M. Felderer, "NLP-assisted software testing: a systematic mapping of the literature," *Information and Software Technology*, vol. 126, 2020, doi: 10.1016/j.infsof.2020.106321.
- [18] A. A. Robisco and J. M. C. Martinez, "Measuring the model risk-adjusted performance of machine learning algorithms in credit default prediction," *Financial Innovation*, vol. 8, no. 1, 2022, doi: 10.1186/s40854-022-00366-1.
- [19] S. Arakelyan, S. Arasteh, C. Hauser, E. Kline, and A. Galstyan, "Bin2vec: learning representations of binary executable programs for security tasks," *Cybersecurity*, vol. 4, no. 1, 2021, doi: 10.1186/s42400-021-00088-4.
- [20] S. Y. Yerima, M. K. Alzaylaee, and S. Sezer, "Machine learning-based dynamic analysis of Android apps with improved code coverage," *Eurasip Journal on Information Security*, vol. 2019, no. 1, 2019, doi: 10.1186/s13635-019-0087-1.
- [21] T. Hai, J. Zhou, N. Li, S. K. Jain, S. Agrawal, and I. B. Dhaou, "Cloud-based bug tracking software defects analysis using deep learning," *Journal of Cloud Computing*, vol. 11, no. 1, 2022, doi: 10.1186/s13677-022-00311-8.
- [22] M. Cui, S. Long, Y. Jiang, and X. Na, "Research of software defect prediction model based on complex network and graph neural network," *Entropy*, vol. 24, no. 10, 2022, doi: 10.3390/e24101373.
- [23] K. Jammalamadaka and N. Parveen, "Testing coverage criteria for optimized deep belief network with search and rescue," *Journal of Big Data*, vol. 8, no. 1, 2021, doi: 10.1186/s40537-021-00453-7.
- [24] N. Laranjeiro, S. N. Soydemir, N. Ivaki, and J. Bernardino, "Testing data-centric services using poor quality data: from relational to NoSQL document databases," *Journal of the Brazilian Computer Society*, vol. 23, no. 1, 2017, doi: 10.1186/s13173-017-0063-x.
- [25] C. L. -Martín, "Machine learning techniques for software testing effort prediction," *Software Quality Journal*, vol. 30, no. 1, pp. 65–100, 2022, doi: 10.1007/s11219-020-09545-8.
- [26] A. Sebastian, H. Naseem, and C. Catal, "Unsupervised machine learning approaches for test suite reduction," *Applied Artificial Intelligence*, vol. 38, no. 1, 2024, doi: 10.1080/08839514.2024.2322336.
- [27] M. H. Moghadam, M. Borg, M. Saadatmand, S. J. Mousavirad, M. Bohlin, and B. Lisper, "Machine learning testing in an ADAS case study using simulation-integrated bio-inspired search-based testing," *Journal of Software: Evolution and Process*, vol. 36, no. 5, 2024, doi: 10.1002/smr.2591.
- [28] F. S. Ahmed, A. Majeed, T. A. Khan, and S. N. Bhatti, "Value-based cost-cognizant test case prioritization for regression testing," *PLoS ONE*, vol. 17, no. 5 May, 2022, doi: 10.1371/journal.pone.0264972.
- [29] R. C. Chen, C. Dewi, S. W. Huang, and R. E. Caraka, "Selecting critical features for data classification based on machine learning methods," *Journal of Big Data*, vol. 7, no. 1, 2020, doi: 10.1186/s40537-020-00327-4.
- [30] Y. Gorishniy, I. Rubachev, V. Khrulkov, and A. Babenko, "Revisiting deep learning models for tabular data," *Advances in Neural Information Processing Systems*, vol. 23, pp. 18932–18943, 2021.
- [31] P. F. Orrù, A. Zoccheddu, L. Sassu, C. Mattia, R. Cozza, and S. Arena, "Machine learning approach using MLP and SVM algorithms for the fault prediction of a centrifugal pump in the oil and gas industry," *Sustainability*, vol. 12, no. 11, 2020, doi: 10.3390/su12114776.
- [32] S. I. Kampeziadou, A. T. Ray, A. P. Bhat, O. J. P. Fischer, and D. N. Mavris, "Fundamental components and principles of supervised machine learning workflows with numerical and categorical data," *Engineering*, vol. 5, no. 1, pp. 384–416, 2024, doi: 10.3390/eng5010021.
- [33] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin, "Catboost: unbiased boosting with categorical features," *Advances in Neural Information Processing Systems*, vol. 2018, pp. 6638–6648, 2018.
- [34] S. Haldar and L. F. Capretz, "Interpretable software defect prediction from project effort and static code metrics," *Computers*, vol. 13, no. 2, 2024, doi: 10.3390/computers13020052.
- [35] H. Zenati, M. Romain, C. S. Foo, B. Lecouat, and V. Chandrasekhar, "Adversarially learned anomaly detection," *IEEE International Conference on Data Mining, ICDM*, pp. 727–736, 2018, doi: 10.1109/ICDM.2018.00088.
- [36] Z. Peng, X. Xiao, G. Hu, A. K. Sangaiah, M. Atiquzzaman, and S. Xia, "ABFL: an autoencoder based practical approach for software fault localization," *Information Sciences*, vol. 510, pp. 108–121, 2020, doi: 10.1016/j.ins.2019.08.077.
- [37] M. Manpreet and J. K. Chhabra, "A hybrid approach based on k-nearest neighbors and decision tree for software fault prediction," *Kuwait Journal of Science*, 2022, doi: 10.48129/kjs.18331.
- [38] K. Barkalov, A. Shtanyuk, and A. Sysoyev, "A fast kNN algorithm using multiple space-filling curves," *Entropy*, vol. 24, no. 6, 2022, doi: 10.3390/e24060767.
- [39] M. Mustaqeem and M. Saqib, "Principal component based support vector machine (PC-SVM): a hybrid technique for software defect detection," *Cluster Computing*, vol. 24, no. 3, pp. 2581–2595, 2021, doi: 10.1007/s10586-021-03282-8.
- [40] Z. Li, Y. Zhao, X. Hu, N. Botta, C. Ionescu, and G. H. Chen, "ECOD: unsupervised outlier detection using empirical cumulative distribution functions," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 12, pp. 12181–12193, 2023, doi: 10.1109/TKDE.2022.3159580.
- [41] X. Du, E. Zuo, Z. Chu, Z. He, and J. Yu, "Fluctuation-based outlier detection," *Scientific Reports*, vol. 13, no. 1, 2023, doi: 10.1038/s41598-023-29549-1.

BIOGRAPHIES OF AUTHORS






Dr. Sangeetha Govinda    is an associate professor in the Department of Computer Science at Christ (Deemed to be University), Central Campus, Bangalore, India, holds a Ph.D. degree from Bharathiar University, Coimbatore. With an extensive career spanning over 19+ years, she has made significant contributions to teaching, research, and administration, shaping educational methodologies across undergraduate and postgraduate levels. Her scholarly endeavors are underscored by the publication of 16 national and 10 international research papers in prestigious journals indexed in IEEE, WoS, and Scopus. She has 5 textbooks and 2 reference books to her credit. Beyond academia, she actively serves as a board of examination (BOE) member for esteemed institutions such as Bengaluru City University, Bangalore University, and Mount Carmel College. Additionally, she contributes her expertise as a member of the review committee for ASTES journal. Her diverse research interests encompass data mining, IoT, software engineering, cryptography, computer networks, R basics, and IT for business. Her dedication to academic excellence extends beyond research, as evidenced by her numerous invited talks, guest lectures, international and national conference participation, and organization of workshops, seminars, and Faculty Development Programs (FDPs). Recognized for her outstanding contributions, she was honored as a Microsoft Research Fellow in 2014. Furthermore, she has received acclaim for her innovative work, including an Australian Patent (No. 2021103341) granted for eight years from June 15, 2021, on August 4, 2021, for her groundbreaking project titled "Artificial intelligence based automatic detection of infection rate of COVID-19." She can be contacted at email: sangeetha.g@christuniversity.in.



Dr. B. G. Prasanthi    works as associate dean and HOD in the Department of Computer Science and Applications at St Joseph's University, Bangalore. She holds degrees of M.Sc., M.Tech., M.Phil., M.B.A., and Ph.D. She has been engaged in innovative teaching practices, resulting in 100% distinctions to all MCA students in subjects like file structures, computer networks, cyber security, and mobile apps. Helped students in taking up online courses. Her research endeavor resulted in developing an embedded chip for CISCO routers. She also believes in continuously enhancing knowledge and works towards the same. She has published 30+ international journals and 20+ national and international conference manuscripts. As a supervisor, she guided research scholars for Ph.D. from Bharathiar University, M.Phil. students for PRIST University, M.Tech., and MCA students for Bangalore University, and MCA, M.Sc. students for St. Joseph's University. She has also been appointed as Deputy Custodian for MCA, M.Sc., computer science for Bangalore University, BOS member Bangalore University, Garden City University, National College, Maharani College, Surana College, Ramaiah College, Jyothi Nivas College, BOE Jain University, National College, Maharani Ammani College, Indian Academy. She was a valuation and external examiner for many universities and a reviewer for many international journals. She can be contacted at email: prasanthi.b.g@sju.edu.in.



Ms. Agnes Nalini Vincent    is the Dean of Faculty of Information Technology and Head of Teaching and Learning at AMITY Institute of Higher Education (AIHE), Mauritius. She holds a Master's in Engineering degree from Anna University, Chennai, India and is currently pursuing her Ph.D. in the field of artificial intelligence. With over 17+ years of experience in teaching, research and administration, she has been instrumental in framing the pedagogies from undergraduate to post graduate studies. She has been the pioneer to develop curriculum in big data analytics, internet of things and data sciences and launched them as courses in Mauritius. She has published national and international research papers in journals indexed in IEEE and Scopus. As a faculty, she has been offering a wealth of talent in the development and implementation of educational technology tools and applications in the classroom. Her area of interests includes artificial intelligence, data mining, big data analytics, internet of things, computer networks, and business data analytics. She has deeply invested in achieving her tenure through administrative service contributions and an accomplishment-oriented approach to teaching. She has given more than 10+ invited talks, 12+ guest lectures, has conducted 1 international conference, has organized 5+ national and international workshops. Her contribution to quality standards formulation and to development of credit system in the capacity of head of teaching and learning at AIHE is remarkable. Her sterling track record of academic excellence and visionary leadership brings a wealth of knowledge, experience, and insight to the forefront of the ICT field. Through her unique and people-centric approach towards administration and management, she has always fostered a supportive and collaborative environment that enables each member of the team to reach their full potential. She can be contacted at email: vanalini@mauritius.amity.edu.