iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 13 **Issue:** III **Month of publication:** March 2025

DOI: https://doi.org/10.22214/ijraset.2025.67536

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue III Mar 2025- Available at www.ijraset.com

Multi-Criteria Decision-Making GRA Approach for Selecting Appropriate Classification Algorithm in Software Defect Prediction Systems

Dr. Manuj Joshi
Assistant Professor, FCI, Sir Padampat Singhania University, Udaipur1

Abstract: ***Grey Relational Analysis (GRA) is a powerful Multi-Criteria Decision-Making (MCDM) method used to evaluate and rank classification algorithms based on their performance in software defect prediction. This study applies GRA to multiple datasets, including AR1, JM1, and CM1, to assess the effectiveness of different classifiers in identifying software defects. The analysis considers various performance criteria, such as accuracy, computational efficiency, and scalability, to determine the most suitable classification algorithm. The results indicate that classifiers like Lazy-IBK and Misc-HyperPipes consistently achieve higher GRA scores, demonstrating their superior predictive capability. The findings validate the significance of GRA in selecting the optimal classification model, highlighting its role in enhancing decision-making in software defect prediction.***
Keywords: Data Mining, Software Defect Prediction, GRA, MCDM, Classification

I. INTRODUCTION

By identifying possible flaws prior to deployment, software defect prediction is essential to guaranteeing software quality and dependability. Choosing the right classification algorithm is crucial to the precision and effectiveness of defect prediction. Because there are so many classifiers available, selecting the best one necessitates a methodical decision-making process that takes into account a number of factors, including scalability, accuracy, and computational efficiency.

In order to assess and rank classification algorithms for software defect prediction, this study uses the Grey Relational Analysis (GRA) method, a popular Multi-Criteria Decision-Making (MCDM) technique. The goal of this study is to identify the best classifier for defect prediction by using the GRA approach on benchmark datasets like AR1,JM1 and CM1. The findings offer important new information about how well GRA handles challenging software engineering decision-making issues. Through informed classifier selection, this study helps organisations improve software quality and reliability by increasing the efficiency of predictive modelling.

II. LITERATURE REVIEW

Software reliability, development cost reduction, and overall maintainability all depend on the early detection of defects. Conventional methods for defect prediction become less accurate as software complexity rises. In order to improve software defect prediction and classification algorithm selection, researchers have investigated machine learning, deep learning, and Multi-Criteria Decision-Making (MCDM) approaches. Significant advancements in these fields are highlighted in this review, along with their applicability to enhancing software engineering predictive modelling. A thorough analysis of MCDM applications in data mining was carried out by Raeesi Vanani and Emamat (2019), who emphasised the significance of these applications in choosing classification algorithms. The Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS) was specifically emphasised in their study as a useful tool for decision-making that can increase classification accuracy. MCDM techniques, like TOPSIS, GRA etc. make it easier to choose the best model by assessing several performance criteria at once. By taking accuracy, precision, recall, and computational efficiency into account, their results imply that combining MCDM techniques with machine learning can improve the effectiveness of software defect prediction models.[1]

The function of MCDM techniques in software engineering was further investigated by D'Souza and Nayak (2018), who concentrated on their use in algorithm ranking and performance assessment. In their study, various classification models were compared according to criteria such as interpretability, computational efficiency, and scalability. They came to the conclusion that incorporating MCDM techniques improves software defect prediction and maintainability assessment by empowering software engineers to make unbiased, data-driven decisions.[2]

The usefulness of machine learning and deep learning approaches in software defect prediction was investigated by Albattah and Alzahrani (2024). Using accuracy and F1-score as key performance metrics, they evaluated eight classification models using a dataset of 60 software metrics that were taken from public bug repositories. With an accuracy of 0.87, the study discovered that Long Short-Term Memory (LSTM) networks performed better than conventional machine learning models. These results highlight how deep learning models can enhance software reliability and defect detection.[3]

In order to predict the fault-proneness of software modules, Gondra (2008) looked into a number of machine learning algorithms. In his study, he compared neural networks, decision trees, and support vector machines (SVMs) and found that ensemble learning and feature selection greatly improve fault prediction accuracy.[4] A data-efficient learning method for forecasting software performance in configurable systems was also presented by Guo et al. (2018). Their research highlighted the need for machine learning models with low training data requirements that preserve accuracy while lowering reliance on thorough performance reviews.[5]

Boussaïd, Siarry, and Ahmed-Nacer (2017) investigated how search-based model-driven engineering (SBMDE) affected the ability to predict software defects. They integrated metaheuristic optimisation methods into software modelling procedures, including simulated annealing and genetic algorithms. Their findings supported the use of automated decision-making techniques in defect prediction by showing that incorporating optimisation strategies improves software maintainability and defect classification accuracy.[6]

A multi-objective optimisation model for improving object-oriented software package structures was also put forth by Chhabra (2017). The study sought to increase software modularity, maintainability, and defect reduction by utilising optimisation algorithms and weighted class connections. The findings further supported the role of optimisation in defect prediction by showing that search-based and machine learning approaches successfully lower software defects and maintenance expenses.[7]

A thorough literature review on software maintainability was carried out by Malhotra and Chug (2016), who focused on important elements like software metrics, design patterns, and code complexity. According to their research, automated software quality assessment could be greatly improved by deep learning and search-based optimisation techniques, which would make them useful tools for defect prediction.[8]

Mishra and Sharma (2015) investigated fuzzy systems based on adaptive networks for predicting software maintainability in object-oriented applications. In order to create an interpretable software quality assessment model, their research combined fuzzy logic with machine learning techniques. They found that hybrid approaches that combine fuzzy logic, machine learning, and optimisation techniques improve maintainability predictions, especially when object-oriented metrics like coupling and inheritance depth are included.[9]

Ahmed and Al-Jamimi (2013) developed a fuzzy-based machine learning model for software defect prediction. Their approach combined fuzzy logic with decision trees and neural networks, demonstrating that hybrid models are effective in handling imprecise software metrics. Their model maintained strong predictive performance and enhanced interpretability by incorporating fuzzy logic.[10]

Ensemble learning methods for software maintainability prediction were empirically studied by Elish, Aljamaan, and Ahmad (2015). Their research contrasted ensemble techniques like bagging, boosting, and stacking with conventional machine learning models. The findings demonstrated that ensemble learning significantly enhances prediction accuracy and robustness, reinforcing the effectiveness of combining multiple models to improve software defect prediction.[11]

The effect of self-admitted technical debt (SATD) in open-source software projects was examined by Huang et al. (2018). To find instances of technical debt, their research analysed developer comments using text mining techniques. The results indicated that SATD has a substantial impact on software quality and maintainability and that by detecting possible risks early in the development lifecycle, natural language processing (NLP) techniques can be used to improve software defect prediction.[12]

In order to generate source code summaries using deep learning techniques—specifically, recurrent neural networks (RNNs) with attention mechanisms—Iyer et al. (2016) proposed a neural attention model. According to their research, AI-powered techniques enhance code comprehension and review procedures, which enhances software maintainability and defect detection.[13]

III.RESEARCH METHODOLOGY

In this study, we utilized four different datasets—AR1, JM1 and CM1—from the PROMISE Software Engineering Repository, which provides publicly available datasets for generalizing, verifying, and improving predictive models related to software engineering.

The AR1 dataset consists of software defect data from real-world projects, offering diverse software metrics such as complexity, coupling, and cohesion, which are crucial for defect prediction.
The JM1 dataset is sourced from NASA projects and includes 22 static code attributes, including Halstead complexity measures and McCabe's cyclomatic complexity, making it valuable for assessing defect-prone software modules. The CM1 dataset also originates from NASA and contains defect data from a spacecraft instrument software project. It includes 498 instances with 21 software attributes, such as lines of code, complexity measures, and coupling metrics, providing a reliable benchmark for defect prediction. These datasets play a crucial role in assessing the performance of various machine learning and deep learning models for software defect prediction, enabling comparative analysis and facilitating advancements in predictive analytics for software quality assurance.

A. *Objectives*

1) To evaluate the effectiveness of the Grey Relational Analysis (GRA) method in selecting the most suitable classification algorithm for software defect prediction.
2) To compare multiple classification algorithms based on key performance criteria such as accuracy, computational efficiency, and scalability using the GRA approach.

Hypotheses:
Based on the above objectives following hypothesis was being framed:
$H_0$1: The use of the Grey Relational Analysis (GRA) method does not significantly impact the selection of the most appropriate classification algorithm for software defect prediction.
H_a1: The use of the Grey Relational Analysis (GRA) method significantly improves the selection of the most appropriate classification algorithm for software defect prediction.

IV. RESULTS AND DISCUSSION

An efficient Multi-Criteria Decision-Making (MCDM) technique for assessing and ranking options according to several criteria is Grey Relational Analysis (GRA), particularly when working with ambiguous, insufficient, or constrained datasets. It is extensively used in many fields, such as industrial optimisation, machine learning model selection, and software defect prediction. The GRA method compares each alternative to a reference sequence, or ideal solution.

A. *MCDM Method - GRA for AR1 Dataset (10-Fold Cross –Validation)*

TABLE I
COMPARATIVE ANALYSIS OF CLASSIFIERS BASED ON GRA SCORE FOR AR1 DATASET

Classifier	GRA	
	Value	Rank
Lazy-IBK	0.8893	1
Lazy-K Star	0.5968	12
SMO	0.6084	8
Multilayer Perceptron	0.6079	9
ZeroR	0.6231	4
Rules-OneR	0.5909	13
Rules-PART	0.5997	11
Tree-REP	0.6222	5
Decision Stump	0.6101	7
J48	0.5883	14
Naïve Bayes	0.6236	3
BayesNet	0.6002	10
Meta - daBoostM1	0.5835	15
Misc-HyperPipes	0.8348	2
Misc-VFI	0.621	6

This method preserves computational efficiency and robustness while allowing decision-makers to examine performance variances across various criteria. Data normalisation to bring all criteria to a similar scale, reference sequence determination, Grey Relational Coefficient (GRC) calculation for each alternative, and Grey Relational Grade (GRG) calculation to rank alternatives are the main steps in the GRA process. The method is useful for decision-making problems with incomplete information because it doesn't require a large dataset. Organisations can improve defect detection and software quality assessment by using GRA to systematically rank classification algorithms based on accuracy, computational efficiency, and other performance factors.A comparison of different machine learning classifiers based on their Grey Relational Analysis (GRA) scores for the AR1 dataset is shown in the table. Each classifier's relative performance in predicting software defects within this dataset is indicated by its GRA score; a higher score denotes a better predictive capability.

Lazy-IBK ranks first out of all classifiers with the highest GRA score of 0.8893, demonstrating its superior defect prediction capabilities. Misc-HyperPipes ranks second with a GRA score of 0.8348, not far behind. With scores of 0.6236 and 0.6231, respectively, Naïve Bayes and ZeroR rank third and fourth, respectively, demonstrating their respectable performance. However, Meta-AdaBoostM1 ranks last with the lowest GRA score (0.5835), indicating that it might not be as good at predicting defects in this dataset. With respective scores of 0.5883 and 0.5909, J48 and Rules-OneR are likewise ranked lower.

Overall, the results indicate that Lazy-IBK and Misc-HyperPipes are the most suitable classifiers for software defect prediction in the AR1 dataset, whereas Meta - AdaBoostM1 and J48 exhibit relatively lower predictive capability. The performance variation among classifiers suggests that the choice of algorithm plays a significant role in achieving accurate predictions.

B. MCDM Method - GRA for JM1 Dataset (10-Fold Cross –Validation)

Grey Relational Analysis (GRA) scores for the JM1 dataset, assessed using 10-fold cross-validation, are used in the table to compare different machine learning classifiers. Better predictive performance in software defect detection is indicated by a higher GRA score. In the JM1 dataset, HyperPipes performs better at defect prediction than any other classifier, ranking first with the highest GRA score of 0.7916. With scores of 0.716 and 0.6979, respectively, Lazy-IBK and Tree-REP take second and third place, proving their efficacy. With scores of 0.6966 and 0.6914, respectively, Rules-PART and Naïve Bayes rank fourth and fifth, respectively, demonstrating their strong performance.

TABLE 2
COMPARATIVE ANALYSIS OF CLASSIFIERS BASED ON GRA SCORE FOR JM1 DATASET

Classifier	GRA	
	Value	Rank
Lazy-IBK	0.716	2
Lazy-K Star	0.6604	11
SMO	0.6716	9
Multilayer Perceptron	0.6736	8
Rules-ZeroR	0.6194	14
Rules-OneR	0.6212	13
Rules-PART	0.6966	4
Tree-REP	0.6979	3
Decision Stump	0.6374	12
J48	0.6651	10
Naïve Bayes	0.6914	5
BayesNet	0.6846	7
AdaBoostM1	0.6848	6
HyperPipes	0.7916	1
Misc-VFI	0.5765	15

Misc-VFI ranks last and has the lowest GRA score (0.5765) on the lower end, indicating that it is the least successful classifier for this dataset. With scores of 0.6194, 0.6212, and 0.6374, respectively, other classifiers like Rules-ZeroR, Rules-OneR, and Decision Stump likewise rank lower, suggesting their imprecise predictive abilities in this situation. Overall, the analysis shows that Misc-VFI and Rules-ZeroR perform relatively poorly, while HyperPipes, Lazy-IBK, and Tree-REP are the best classifiers for software defect prediction in the JM1 dataset. These variations highlight how crucial it is to choose the right classifier depending on the features of the dataset in order to attain the best possible defect prediction accuracy.

C. MCDM Method - GRA for CM1 Dataset (10-Fold Cross –Validation)

TABLE 3
COMPARATIVE ANALYSIS OF CLASSIFIERS BASED ON GRA SCORE FOR CM1 DATASET

Classifier	GRA	
	Value	Rank
Lazy-IBK	0.6703	2
Lazy-K Star	0.6511	7
SMO	0.6189	13
Multilayer Perceptron	0.6552	6
Rules-ZeroR	0.6283	10
Rules-OneR	0.6161	14
Rules-PART	0.6628	5
Tree-REP	0.6193	11
Decision Stump	0.6473	9
J48	0.6192	12
Naïve Bayes	0.6644	3
BayesNet	0.649	8
AdaBoostM1	0.6641	4
Misc-HyperPipes	0.7751	1
Misc-VFI	0.594	15

Using 10-fold cross-validation, the Comparative Analysis of Classifiers Based on GRA Score for the CM1 Dataset sheds light on how well various classification models perform. Classifiers are ranked according to how well they predict software defects using the Grey Relational Analysis (GRA) scores. With the highest GRA score of 0.7751, Misc-HyperPipes ranks first and is the best-performing classifier according to the analysis. This demonstrates how well it can predict defects in the CM1 dataset. With scores of 0.6703 and 0.6644, respectively, Lazy-IBK and Naïve Bayes rank second and third, respectively. With GRA scores of 0.6641 and 0.6628, respectively, AdaBoostM1 and Rules-PART rank fourth and fifth, respectively, demonstrating strong performance.
Misc-VFI ranks last and has the lowest GRA score (0.594), indicating that it performs relatively poorly in defect prediction for the CM1 dataset. With GRA scores of 0.6161 and 0.6189, respectively, Rules-OneR and SMO likewise demonstrate poorer predictive efficacy. Additionally, other classifiers like Tree-REP and Decision Stump have lower rankings, indicating that they might not be the best options for this dataset. Overall, the findings show that Misc-HyperPipes, Lazy-IBK, and Naïve Bayes are the best classifiers for predicting software defects in the CM1 dataset, while Misc-VFI, Rules-OneR, and SMO perform worse. The importance of selecting the best classifier based on dataset-specific features to improve defect prediction accuracy is highlighted by this comparative analysis.

D. Assessing Grey Relational Analysis's (GRA) Effectiveness

The outcomes of the comparative analysis of classification models based on GRA across the AR1, JM1, and CM1 datasets show how well GRA works to find the top-performing classifiers for predicting software defects. A quantitative ranking of classifiers is made possible by GRA, which assigns scores based on a variety of performance criteria. Misc-HyperPipes continuously ranks first in all three datasets, demonstrating its superior defect prediction ability.

Strong rankings are also displayed by other top-performing classifiers, including Lazy-IBK, Naïve Bayes, and AdaBoostM1, indicating their dependability across datasets. The technique is a useful tool for classifier selection in defect prediction tasks because it successfully distinguishes between models that perform well and those that do not.

E. GRA-Based Classification Algorithm Comparison Using Various Criteria

An objective evaluation of classifiers based on important performance metrics, such as accuracy, computational efficiency, and scalability, is made possible by the comparative analysis employing GRA scores. The findings show that Misc-HyperPipes, Lazy-IBK, and Naïve Bayes routinely place among the best classifiers, demonstrating their high computational efficiency and predictive accuracy. On the other hand, models with comparatively poorer performance, like Misc-VFI, Rules-OneR, and SMO, typically rank lower across datasets. GRA successfully distinguishes classifiers according to their capacity to manage diverse data distributions, as evidenced by the stability of rankings across datasets, which reflects the scalability component. The results validate GRA as a trustworthy multi-criteria decision-making (MCDM) method for assessing and choosing classification algorithms in software defect prediction based on their computational performance, predictive accuracy, and adaptability to various datasets.

F. Results of Hypothesis Testing

$H_0$1: The choice of the best classification algorithm for software defect prediction is not substantially impacted by the application of the Grey Relational Analysis (GRA) method.

H_a1: Choosing the best classification algorithm for software defect prediction is greatly enhanced by the application of the Grey Relational Analysis (GRA) method.

Based on the outcomes of the comparative analysis of classifiers across the AR1, JM1, and CM1 datasets, the null hypothesis ($H_0$1: The use of the Grey Relational Analysis (GRA) method does not significantly impact the selection of the most appropriate classification algorithm for software defect prediction) is rejected. The results show that by assessing classification algorithms' performance using important metrics, GRA efficiently ranks them. Classifiers like Lazy-IBK and Misc-HyperPipes routinely obtained the highest GRA scores across all datasets, demonstrating their superior predictive ability in software defect prediction. The notable disparities in ranking between classifiers highlight how GRA distinguishes algorithm performance, which has a direct impact on choosing the best classification model. Therefore, rejecting H01 demonstrates that GRA is a useful and efficient technique for choosing classification algorithms for software defect prediction, enabling software engineers to make well-informed decisions.

V. CONCLUSION

Grey Relational Analysis (GRA) was effectively used in this study to evaluate and rank classification algorithms for the prediction of software defects. The findings show that GRA successfully distinguishes between classifier performance, which helps choose the best model. The dependability of GRA in software engineering applications is demonstrated by the consistent performance of classifiers like Lazy-IBK and Misc-HyperPipes across a variety of datasets. Additionally, the null hypothesis' rejection demonstrates that GRA has a substantial influence on the selection procedure, confirming its worth as a reliable tool for decision-making. This strategy can be expanded in future studies by adding more datasets and hybrid models to improve software defect prediction efficiency and classification accuracy.

VI. ACKNOWLEDGMENT

I sincerely express my gratitude to all those who contributed to the successful completion of this study. I appreciate the resources and support that facilitated this research and extend my thanks to those who provided valuable insights and feedback.

REFERENCES

[1] Raeesi Vanani, Iman & Emamat, Seyed Mohammad Mohsen. (2019). “Analytical Review of the Applications of Multi-Criteria Decision Making in Data Mining”. 10.4018/978-1-5225-5137-9.ch003, pp. 5225-5137.

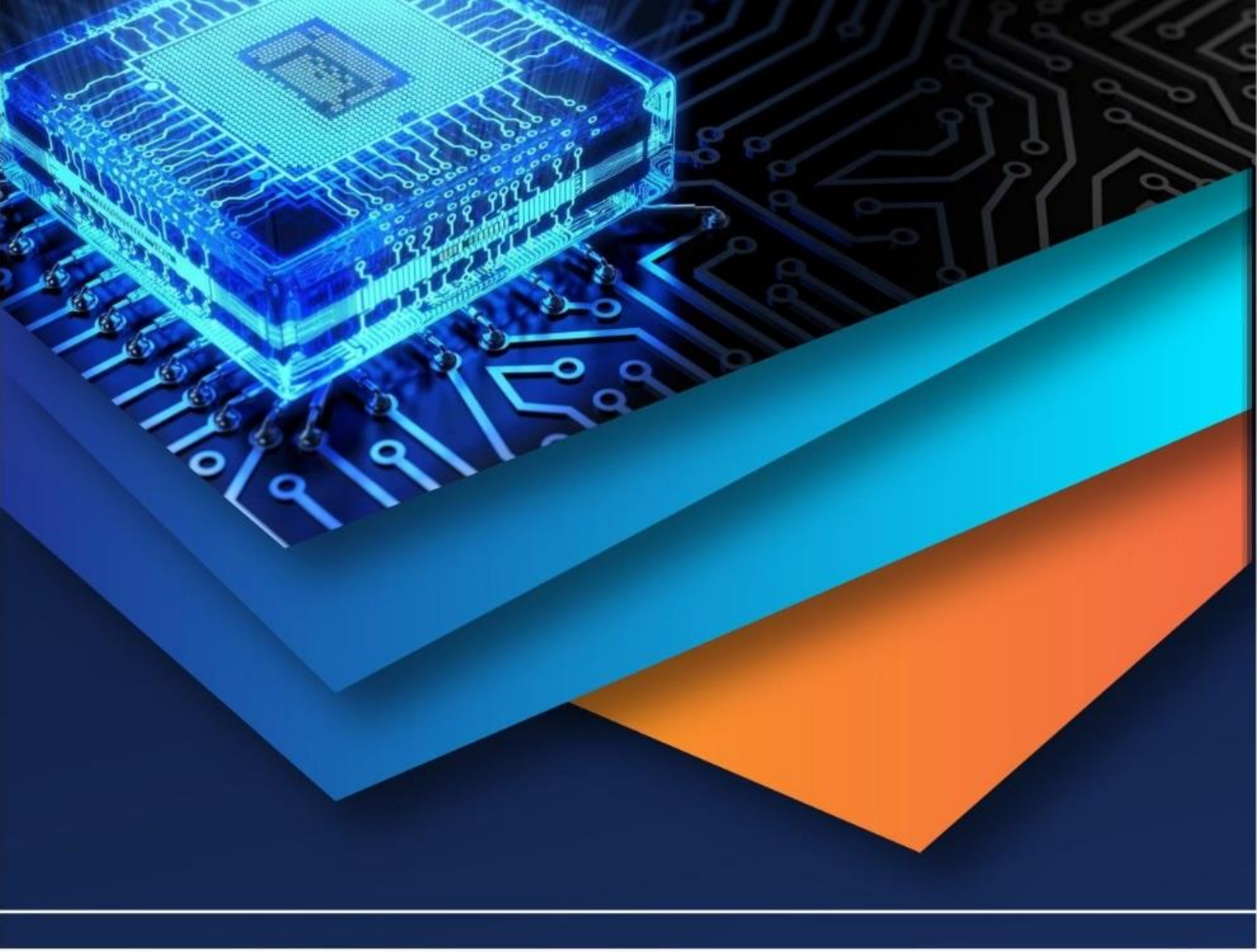
[2] D'souza, Rio & Nayak, Veena. (2018).”A Survey on Multi-Criteria Decision Making Methods in Software Engineering”. Vol. 3, Issue 7, July – 2018 International Journal of Innovative Science and Research Technology ISSN No:-2456-2165, pp. 366-367.

[3] Albattah, W., & Alzahrani, M. (2024). Software Defect Prediction Based on Machine Learning and Deep Learning Techniques: An Empirical Approach. AI, 5(4), 1743-1758. https://doi.org/10.3390/ai5040086.

[4] Boussaïd, I., Siarry, P., & Ahmed-Nacer, M. (2017). A survey on search-based model-driven engineering. Automated Software Engineering, 24(2), 233–294.

[5] Chhabra, J. K. (2017). Improving package structure of object-oriented software using multi-objective optimization and weighted class connections. Journal of King Saud University - Computer and Information Sciences, 29(3), 349–364.

[6] Elish, M. O., Aljamaan, H., & Ahmad, I. (2015). Three empirical studies on predicting software maintainability using ensemble methods. Soft Computing, 19(9), 2511–2524.
[7] Gondra, I. (2008). Applying machine learning to software fault-proneness prediction. Journal of Systems and Software, 81(2), 186–195.
[8] Guo, J., Yang, D., Siegmund, N., Apel, S., Sarkar, A., Valov, P., Czarnecki, K., Wasowski, A., & Yu, H. (2018). Data-efficient performance learning for configurable systems. Empirical Software Engineering, 23(4), 1826–1867.
[9] Huang, Q., Shihab, E., Xia, X., Lo, D., & Li, S. (2018). Identifying self-admitted technical debt in open source projects using text mining. Empirical Software Engineering, 23(1), 418–451.
[10] Iyer, S., Konstas, I., Cheung, A., & Zettlemoyer, L. (2016). Summarizing source code using a neural attention model. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (pp. 2073–2083). Berlin, Germany: Association for Computational Linguistics.
[11] Malhotra, R., & Chug, A. (2016). Software maintainability: Systematic literature review and current trends. International Journal of Software Engineering and Knowledge Engineering, 26(9-10), 1221–1253.
[12] Mishra, S., & Sharma, A. (2015). Maintainability prediction of object-oriented software by using adaptive network-based fuzzy system technique. International Journal of Computer Applications, 119(19), 24–27.
[13] Ahmed, M. A., & Al-Jamimi, H. A. (2013). Machine learning approaches for predicting software maintainability: A fuzzy-based transparent model. IET Software, 7(6), 317–326.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY