International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 09 Issue: 03 | March - 2025 SJIF Rating: 8.586 ISSN: 2582-3930

Applying TOPSIS Method for Optimal Data Mining Algorithm Selection in Software Defect Prediction Systems

Dr. Manuj Joshi[1]

[1]*Assistant Professor, FCI, Sir Padampat Singhania University, Udaipur*

Abstract - In software defect prediction systems, choosing the best data mining classification algorithm is essential for efficient decision-making. The Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS) method for ranking classification algorithms according to several criteria is assessed in this study. Classifiers were evaluated and ranked using the AR1 and JM1 datasets using a structured multi-criteria decision-making (MCDM) methodology. According to the results, Lazy-IBK continuously performs better than other classifiers, demonstrating how well TOPSIS finds the top-performing models. Furthermore, the comparative study of classification algorithms shows how reliable the approach is when taking into account variables like scalability, accuracy, and computational efficiency. According to the results, TOPSIS considerably enhances classifier selection when it comes to software defect prediction, resulting in data mining applications that make better-informed and impartial decisions.

***Key Words*:** MCDM, Software Defect Prediction, TOPSIS

1.INTRODUCTION

Organizations use data mining classification algorithms to glean insightful information and make well-informed decisions in today's data-driven world. The process of choosing the best classification algorithm is difficult since it takes into account a number of factors, including scalability, interpretability, accuracy, and computational efficiency. Methods known as Multi-Criteria Decision-Making (MCDM) offer an organized way to assess and rank these algorithms according to a number of performance metrics. Because it ranks alternatives by comparing them to an ideal and a negative-ideal solution, the Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS) is one of the most popular MCDM techniques. By taking into account both the best and worst-case scenarios, TOPSIS guarantees a fair assessment of classification algorithms according to predetermined standards.

The impact of the TOPSIS method on choosing the best data mining classification algorithm in organizational information systems is investigated in this study. Organizations can improve decision-making efficiency, priorities different classification models according to business goals, and evaluate them methodically by implementing TOPSIS. The goal of the study is to demonstrate how the approach enhances model selection, boosts accuracy, and improves information systems decision support generally.

2. LITERATURE REVIEW

The Early software defect detection is essential for increasing software dependability and cutting development expenses. With a growing emphasis on Multi-Criteria Decision Making (MCDM) techniques for algorithm selection and optimisation, numerous studies have investigated machine learning and deep learning approaches for software defect prediction.

In their thorough analysis of MCDM applications in data mining, Raeesi Vanani and Emamat (2019) emphasised their function in choosing the best classification algorithms. By assessing several performance criteria at once, the TOPSIS method improves classification accuracy and is a useful tool for decision-making, according to the study. Their study backs up the use of MCDM techniques to enhance data mining model selection and algorithm efficiency.[1]

In their 2018 study, D'souza and Nayak examined the use of MCDM techniques in software engineering for algorithm ranking and performance assessment. According to their findings, TOPSIS and related methods facilitate the objective comparison of various classification models according to criteria like computational efficiency, scalability, and accuracy. Software engineers can improve defect prediction and maintainability evaluation by optimising model selection through the use of MCDM techniques.[2]

The usefulness of deep learning and machine learning models in software defect prediction was investigated by Albattah and Alzahrani (2024). Based on classification performance metrics like accuracy and F1 scores, the study assessed eight distinct models using a dataset that included 60 software metrics from open bug repositories. Long short-term memory (LSTM) models performed better than other methods, according to the results, with an accuracy of 0.87. This highlights the potential of deep learning techniques to enhance software reliability and defect detection.[3]

By integrating optimisation techniques into software modelling, Boussaïd, Siarry, and Ahmed-Nacer (2017) investigated search-based model-driven engineering (SBMDE). The study evaluated the effect of metaheuristic algorithms on software model structure refinement, including simulated annealing and genetic algorithms. Their study showed that by facilitating more efficient classification and decision-making procedures, the integration of optimisation strategies improves software maintainability and defect prediction.[4]

A multi-objective optimisation model was put forth by Chhabra (2017) with the goal of enhancing object-oriented software package structures. The study aimed to improve software modularity and maintainability by utilising optimisation algorithms and weighted class connections. The findings showed that by optimising software architecture, search-based and machine learning approaches successfully lower software defect rates and maintenance expenses.[5]

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 results showed that ensemble learning is a dependable technique for predicting software defects and

evaluating maintainability since it greatly improves prediction accuracy and robustness.[6]

The use of machine learning algorithms to forecast the fault-proneness of software modules was examined by Gondra (2008). Support vector machines (SVMs), decision trees, and neural networks were among the classification models that were compared in the study. The findings demonstrated that feature selection and ensemble learning strategies increase fault prediction accuracy, underscoring the significance of automated defect prediction systems.[7]

A data-efficient learning method for forecasting software performance in configurable systems was presented by Guo et al. (2018). The study focused on how machine learning models can predict system behaviour with little training data, negating the need for thorough performance reviews. The study emphasised the importance of sampling and feature selection strategies in creating flexible machine learning models for defect prediction.[8]

The problem of self-admitted technical debt (SATD) in open-source software projects was the main focus of Huang et al. (2018). In order to find instances where developers acknowledged technical debt in code comments, the study employed text mining techniques. The results showed that SATD has a major impact on software quality and maintainability, and that by detecting latent software problems early in the development cycle, natural language processing (NLP) techniques can improve software defect prediction models.[9]

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 the study, AI-driven techniques enhance code comprehension and review procedures, which enhances software maintainability and defect prediction.[10]

In their systematic review of the literature on software maintainability, Malhotra and Chug (2016) emphasised important elements like software metrics, design patterns, and code complexity. The study examined a number of statistical and machine learning models and suggested that search-based optimisation and deep learning could improve automated software quality assessment even more.[11]

Mishra and Sharma (2015) investigated the prediction of software maintainability in object-oriented applications using fuzzy systems based on adaptive networks. The study created an interpretable software quality assessment model by combining machine learning and fuzzy logic techniques. According to the findings, hybrid machine learning techniques enhance maintainability predictions, especially when object-oriented metrics like coupling and inheritance depth are included.[12]

A fuzzy-based machine learning model was created by Ahmed and Al-Jamimi (2013) to forecast software maintainability. Their method produced an interpretable prediction system by combining fuzzy logic with decision trees and neural networks. The study demonstrated the potential of fuzzy-based approaches in software defect prediction by highlighting their benefits in handling imprecise and ambiguous software metrics.[13]

The literature review highlights the increasing importance of optimisation, deep learning, and machine learning methods in predicting software defects and assessing maintainability. Hybrid models, feature selection tactics, and advanced AI techniques have shown encouraging results in improving software quality. By combining fuzzy logic, search-based optimisation, and natural language processing for automated software quality assessment, future studies can further enhance predictive models.

3. RESEARCH METHODOLOGY

The impact of the TOPSIS method in choosing the best data mining classification algorithm for software defect prediction is assessed in this study using a quantitative research methodology. Classification algorithms like Decision Tree, Support Vector Machine, Naïve Bayes, Random Forest, K-Nearest Neighbours, and Artificial Neural Networks will be taken into consideration, and publicly accessible benchmark datasets or real-world organisational datasets will be used. Decision criteria will be based on key evaluation metrics, such as scalability, accuracy, precision, recall, F1-score, and computational efficiency. The algorithms will be ranked using the TOPSIS method, which compares them to both an ideal and a negative-ideal solution. The findings will be examined to determine how TOPSIS enhances decision-making in software defect prediction systems and influences algorithm selection.

Objectives:

1. To analyse the effectiveness of the TOPSIS method in selecting the optimal data mining classification algorithm for organizational decision-making.
2. To compare classification algorithms based on multiple criteria, such as accuracy, computational efficiency, and scalability, using the TOPSIS method.

Hypotheses:

Based on the above objectives following hypotheses was being framed:

$H_0$1: The use of the TOPSIS method does not significantly impact the selection of the most appropriate data mining classification algorithm in organizational information systems.

H_a1: The use of the TOPSIS method significantly improves the selection of the most appropriate data mining classification algorithm in organizational information systems.

4. RESULTS AND DISCUSSION

The Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS) is a multi-criteria decision-making (MCDM) method used to rank alternatives based on their distance from an ideal and a negative-ideal solution. It evaluates multiple criteria simultaneously, ensuring an objective and structured selection process. The method calculates a relative closeness score for each alternative, identifying the most optimal choice. TOPSIS is widely used in various domains, including data mining, software defect prediction, and organizational decision-making.

The JM1 dataset is a software defect prediction dataset from NASA's Metrics Data Program (MDP), containing 10,885 instances and 22 attributes, primarily used for predicting software defects to enhance software reliability. It is publicly available through repositories like the PROMISE Software Engineering Repository. The AR1 dataset, on the other hand, is a software defect prediction dataset collected from a Turkish White-Goods manufacturer and was donated by the Software Research Laboratory (Softlab). Both datasets serve as valuable

benchmarks for evaluating machine learning models in software defect prediction.

4.1 MCDM Method - TOPSIS for AR1 Dataset (10-Fold Cross –Validation):

The TOPSIS evaluation for the AR1 dataset using 10-fold cross-validation reveals that the Lazy-IBK classifier performs the best, achieving the highest TOPSIS value of 0.9239 and securing the top rank. This indicates that Lazy-IBK is the most effective classifier in terms of similarity to the ideal solution, demonstrating superior predictive capabilities for this dataset. The SMO classifier follows in second place with a TOPSIS score of 0.51, while Lazy-K Star ranks third with 0.495. These results suggest that instance-based learning methods (Lazy-IBK and Lazy-K Star) perform well, along with support vector-based approaches (SMO). The strong ranking of these classifiers implies that they are well-suited for the specific characteristics of the AR1 dataset.

Table 4.1: Comparative Analysis of Classifiers based on TOPSIS Score for AR1 Dataset

Classifier	TOPSIS	
	Value	Rank
Lazy-IBK	0.9239	1
Lazy-K Star	0.495	3
SMO	0.51	2
Multilayer Perceptron	0.4664	7
ZeroR	0.4435	11
Rules-OneR	0.4747	6
Rules-PART	0.4836	4
Tree-REP	0.4491	9
Decision Stump	0.4276	12
J48	0.4664	8
Naïve Bayes	0.4777	5
BayesNet	0.4201	13
Meta - daBoostM1	0.444	10
Misc-HyperPipes	0.1625	15
Misc-VFI	0.2757	14

Among the moderately performing classifiers, Naïve Bayes (0.4777, Rank 5) and Rules-PART (0.4836, Rank 4) exhibit competitive results, making them viable alternatives in certain scenarios. Multilayer Perceptron (0.4664, Rank 7) and J48 (0.4664, Rank 8) show similar performance, indicating that decision tree and neural network-based models have comparable effectiveness in this dataset. On the lower end, BayesNet (0.4201, Rank 13) and Misc-HyperPipes (0.1625, Rank 15) show significantly weaker performance, suggesting their limited applicability for AR1 dataset classification. Overall, the results emphasize the dominance of Lazy-IBK and SMO, making them the most suitable choices for classification tasks on this dataset.

4.2 MCDM Method - TOPSIS for JM1 Dataset (10-Fold Cross –Validation)

The TOPSIS evaluation for the JM1 dataset using 10-fold cross-validation indicates that Lazy-IBK (0.8013, Rank 1) is the most effective classifier, closely followed by Lazy-K Star (0.7987, Rank 2). These results highlight the dominance of instance-based learning methods for this dataset, suggesting that these classifiers efficiently capture patterns within JM1. SMO (0.51, Rank 3) also demonstrates strong performance, making it a viable choice for classification tasks. The Naïve Bayes classifier (0.3965, Rank 4) outperforms several rule-based and tree-based classifiers, suggesting that probabilistic approaches are relatively effective for the JM1 dataset.

Table 4.2: Comparative Analysis of Classifiers based on TOPSIS Score for JM1 Dataset

Classifier	TOPSIS	
	Value	Rank
Lazy-IBK	0.8013	1
Lazy-K Star	0.7987	2
SMO	0.51	3
Multilayer Perceptron	0.3352	10
Rules-ZeroR	0.294	13
Rules-OneR	0.3604	9
Rules-PART	0.3622	7
Tree-REP	0.3688	6
Decision Stump	0.3108	12
J48	0.3758	5
Naïve Bayes	0.3965	4
BayesNet	0.3611	8
AdaBoostM1	0.3181	11
HyperPipes	0.1934	15
Misc-VFI	0.2494	14

In the mid-range, J48 (0.3758, Rank 5), Tree-REP (0.3688, Rank 6), and Rules-PART (0.3622, Rank 7) exhibit moderate classification ability, indicating that decision tree-based techniques perform fairly well. However, Multilayer Perceptron (0.3352, Rank 10) and AdaBoostM1 (0.3181, Rank 11) show relatively lower effectiveness, implying that neural networks and boosting techniques may not be the best-suited models for this dataset. On the lower end, Rules-ZeroR (0.294, Rank 13), HyperPipes (0.1934, Rank 15), and Misc-VFI (0.2494, Rank 14) exhibit weak classification performance, confirming their limitations for JM1. Overall, the results emphasize that Lazy-IBK and Lazy-K Star remain the top-performing classifiers, making them the most suitable choices for classification tasks on JM1.

4.3 Effectiveness of the TOPSIS Method

The results obtained from the AR1 and JM1 datasets demonstrate the effectiveness of the TOPSIS (Technique for Order Preference by Similarity to Ideal Solution) method in ranking classification algorithms for software defect prediction. By considering multiple criteria, such as accuracy and computational performance, TOPSIS provides a systematic approach to identifying the most suitable classifier for organizational decision-making. In both datasets, Lazy-IBK consistently ranked highest, indicating its strong predictive capability in software defect classification tasks. This reinforces the utility of the TOPSIS method in real-world applications where organizations must select optimal models based on multiple performance measures.

4.4 Comparison of Classification Algorithms Using TOPSIS Based on Multiple Criteria

The comparative analysis of classification algorithms using the TOPSIS method highlights significant variations in their performance. For both datasets, Lazy-IBK and Lazy-K Star emerged as top-ranked classifiers, demonstrating superior predictive accuracy. SMO also performed well, securing a high rank in both datasets. Conversely, algorithms like HyperPipes and Misc-VFI ranked lowest, indicating lower suitability for defect prediction. The rankings emphasize the importance of considering multiple criteria, as some classifiers may excel in accuracy but lag in computational efficiency or scalability. The results validate the applicability of TOPSIS in evaluating classifiers, offering a structured and quantitative approach to decision-making in organizational data mining tasks.

4.5 Hypothesis Testing Results

H01: The use of the TOPSIS method does not significantly impact the selection of the most appropriate data mining classification algorithm in organizational information systems.

Ha1: The use of the TOPSIS method significantly improves the selection of the most appropriate data mining classification algorithm in organizational information systems.

The results from the AR1 and JM1 datasets demonstrate that the use of the TOPSIS method significantly improves the selection of the most appropriate data mining classification algorithm in organizational information systems, leading to the rejection of the null hypothesis (H01) and acceptance of the alternative hypothesis (Ha1). The rankings generated by TOPSIS provide a structured approach to evaluating classifiers based on multiple criteria, ensuring a more balanced and informed decision-making process. For example, Lazy-IBK consistently emerges as the top-performing classifier across both datasets, highlighting the robustness of the TOPSIS ranking in identifying optimal models.

Furthermore, the application of TOPSIS enables the simultaneous consideration of multiple performance factors, such as accuracy, computational efficiency, and scalability, rather than relying solely on traditional accuracy-based measures. This suggests that organizations can leverage TOPSIS to make more data-driven and objective classifier selections, reducing biases and improving decision-making efficiency. The consistent ranking patterns across datasets reinforce the reliability of TOPSIS in classifier selection, making it a valuable tool for optimizing data mining processes in organizational information systems.

5. CONCLUSIONS

The study demonstrates that TOPSIS is a very successful technique for choosing the best classification algorithm for predicting software defects. The dependability of the TOPSIS method is demonstrated by the rankings derived from the AR1 and JM1 datasets, which consistently show that Lazy-IBK performs better than other classifiers. TOPSIS offers a systematic and impartial framework for decision-making by integrating multiple performance criteria, which lessens biases and improves classifier selection accuracy. Furthermore, TOPSIS considerably enhances classifier evaluation in comparison to conventional selection techniques, as further supported by the rejection of the null hypothesis (H01). Additionally, this study highlights the useful benefits of multi-criteria decision-making (MCDM) approaches in data mining applications. Businesses that choose classifiers that strike a balance between accuracy, scalability, and computational performance can use TOPSIS to improve predictive modelling and decision-making effectiveness. The results demonstrate how MCDM can be used to optimise classifier selection, opening the door for further study into hybrid strategies that combine TOPSIS with deep learning models or other MCDM techniques to enhance classifier selection in challenging decision-making situations.

ACKNOWLEDGEMENT

I sincerely appreciate the resources and support that facilitated this research. The access to relevant datasets and research materials greatly contributed to the study. Constructive insights helped refine the analysis. Lastly, I acknowledge the motivation that enabled the successful completion of this work.

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.