A Hybrid Approach Entropy - TOPSIS for the Selection of Machine Learning Classifiers for Software Defect Prediction

Miguel-Ángel Quiroz-Martinez[1], *Eduardo-Xavier Moreira-Bermello*[1], *Wilmer-Paul Carrillo-Leon*[1], *Luis-Andy Briones-Peñafiel*[1]

[1] Department Computer Science, Universidad Politécnica Salesiana, Guayaquil, Ecuador

ABSTRACT

The software is developed with much more complex functionalities to meet user requirements. Therefore, it is more vulnerable and subject to defects during the software development life cycle (SDLC). There are many efforts to avoid and re-duce the number of defects, ensure good performance, and achieve defect-free software before releasing it to the market. To minimize the effort and optimize testing using Machine Learning, classifier models can be created to classify and predict which modules of the developed software may or may not be more prone to defects. There is a wide variety of classifier models; however, there is no classifier model that performs better than another in general terms. Various performance metrics can be used using multiple historical data sets to compare and evaluate classifier models. To select the best classifier model, a hybrid approach combining two

multicriteria decision making (MCDM) methods, Entropy and TOPSIS, is proposed. Entropy is used to calculate the criteria weights, and TOPSIS compares and ranks the alternatives. The results show that the proposed hybrid method can make the distribution of weights more reasonable and the selection of other options more efficient.

Keywords: Machine Learning · ReLink Dataset · Software Defect Prediction · TOPSIS · Entropy

INTRODUCTION

Currently, software development is perhaps one of the areas advancing by leaps and bounds over time. Although, software plays a very important role in the world of technology, it has become indispensable in business and daily life, increasingly efficient, equipped with new resources and functionalities to meet users' needs. Consequently, it makes its development much more complex, vulnerable, and subject to defects. Defects (commonly called bugs) are well-known and unavoidable during the software development life cycle (SDLC), which can cause non-compliance with requirements, undesired behaviors, or incorrect results in functionalities.

Most of the defects are introduced as a consequence of the human factor, which is usually produced and found in the different phases of the SDLC, not only in the development phase. During the entire SDLC, more than 50% of the time is spent maintaining quality and reliability to ensure good performance and achieve defect-free software before re-lease, making it exhausting, costly, and time-consuming to test all the complete soft-ware modules. To solve this problem, many researchers have focused their research on the Artificial Intelligence (AI) branch called Machine Learning (ML) and how it can be used to optimize the tests to be performed to shorten testing cycles, reduce effort and increase their efficiency. Software testing, thanks to Machine Learning, can create classifier models to accurately classify and predict which modules of the developed software may be more prone to defects or not prone to defects. Supervised machine learning classifier models can be trained from one or more historical software defect data sets, using data mining to prepare and extract patterns be-tween data labeled in the defective or non-defective category and descriptive attributes.

There is a wide variety of Machine Learning classifier models that behave differently depending on the amount and distribution of the data provided, some of which are very efficient and popular such as k nearest neighbors (kNN), Support Vector Machines (SVM), Naïve Bayes (NB), Random Forests (RF), Neural Networks (NN), among others. Despite this, no classifier model performs better than another in a general way, so various performance metrics have emerged to compare and evaluate the classification of classifier

models used for a given problem. For this reason, this paper proposes a hybrid approach that combines two multi-criteria decision making (MCDM) methods: Entropy weight combined with the TOPSIS method. Where Entropy is used to calculate the objective weights of the criteria combined with TOPSIS to compare and rank a finite number of alternatives in order of preference. The rest of the article is structured as follows: Section 2 explores the Preliminaries, Section 3 describes the MCDM Method: Entropy and TOPSIS, Section 4 the Empirical Case Study and its result. Finally, Section 5 sets out conclusions and directions for future work.

PRELIMINARIES: DATASET DESCRPTION

This study used 3 datasets: Apache, Safe, ZXing. All these datasets are obtained from the ReLink project, it was developed by Wu, et al and has been widely used in Software Defect Prediction research. They are based on source code linkage data information deduced from version control. Table 1 shows a description and detail of each dataset.

Table 1. Datasets Description

Dataset	Description	Metrics	Instances	Defect
Apache	Webserver	26	194	98 (50.51%)
Safe	Security	26	56	22 (39.28%)
ZXing	Barcode reader library	26	399	118(29.57)

MACHINE LEARNING CLASSFIER MODELS

Classifiers are supervised machine learning models that identify patterns or trends using a set of labeled input data in order to help the models correctly classify the data and predict the appropriate output label for new instances. A brief description is used for each classification model that was used: k Nearest Neighbors (kNN) is a feature similarity-based model. It consists of storing the features of the classes of the training examples. When classifying a new instance, it compares the similarity with the existing examples and places the new instance in the class that is most similar. Support Vector Machines (SVM) is a model based on constructing in the form of a decision surface an optimal hyperplane that defines the largest margin of linear separation between two or more input data classes. Naïve Bayes (NB) is a model based on a strong assumption of independence between predictors, based on Bayes theorem, which finds the probability that a particular feature is related or not to any

other feature. Random Forests (RF) is a model based on the combination of a large number of independent decision trees operating as a forest. Within this forest, each decision tree gives a prediction of the class, and the class with the most "votes" is the prediction of the model. Neural networks (NN) are a model based on the functioning of the nervous system, formed by a set of neurons and brain connections, where the ability to memorize and associate facts helps to solve problems such as complex classification tasks.

PERFORMANCE EVALUATION METRICS

A confusion matrix, also called error matrix or contingency table, is normally used to obtain values such as True Positive (TP), True Negative (TN), False Positive (FP), False Negative (FN), False Positive (FP), and False Negative (FN). See, TP is the amount that "Defaults" were correctly predicted as such, TN is the amount that "No Defaults" were correctly predicted as such. In return, FP is the amount that "No Default" was incorrectly predicted as "Default", FN is the amount that "Default" was incorrectly predicted as "No Default". From the values obtained by the confusion matrix, performance metrics can be calculated, which are statistical techniques used to evaluate the quality and performance of machine learning classifier models. In general, the metrics used are as follows: Accuracy, Precision, Recall, Specificity, F-Measure.

MCDM METHOD: ENTROPY AND TOPSIS

Multicriteria decision-making MCDM methods have been widely used to select the best alternative among a set of decision alternatives characterized by multiple conflicting criteria. Currently, several MCDM methods are available; this section presents the TOPSIS method for evaluating alternatives and the entropy method for determining the weights of each criterion.

ENTROPY METHOD

Shannon entropy is a wellknown method used to accurately determine the weighting based on the objective information of the criteria identified in MCDA problems, which can efficiently reflect the essence of the information and measure the uncertainty in the useful information of the obtained data. It is calculated with the following steps:

Step 1: Determine the decision matrix and normalize the decision matrix P_ij using:

$$p_{ij} = \frac{x_{ij}}{\sum_{i=1}^{m} x_{ij}}, (\, i = 1, \dots, m; j = 1, \dots, n) \quad \textbf{(1)}$$

Step 2: Calculate the value of the entropy E_j by means of:

$$e_j = -\frac{\sum_{i=1}^{m} p_{ij} \ln(p_{ij})}{\ln(\mathrm{m})}), (\, j = 1, \dots, n) \quad \textbf{(2)}$$

Step 3: Finally, the objective weighting of each criterionw_j is calculated as follows:

$$w_j = \frac{1 - e_j}{\sum_{j=1}^{n}(1 - e_j)}, (\, j = 1, \dots, n) \quad \textbf{(3)}$$

TOPSIS METHOD

TOPSIS (Technique For Order Performance by Similarity to Ideal Solution), developed by Hwang and Yoon, is a widely used, easy-to-compute, algorithmically structured MCDM method for solving decision-making problems. The TOPSIS method selects the best alternative, the one with the shortest distance to the positive ideal solution (PIS) and the longest distance to the perfect negative solution (NIS). The TOPSIS procedure consists of the following steps:

Step 1: Determine the m×n decision matrix D= [xij], which is composed of possible alternatives A= {Ai | i = 1, 2, . . ., m} and evaluation criteria C= {Cj | j = 1, 2, . . ., n}. The vector of weights is composed of the individual weights W= {wj | j = 1, 2, . . ., n} for each criterion, obtained by the entropy method (4).

$$D = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{bmatrix} \quad \textbf{(5)}$$

Step 2: Calculate the normalized decision matrix R =[rij]. The normalized value rij is calculated by:

$$\mathrm{r_{ij}} = \frac{\mathrm{x_{ij}}}{\sqrt{\Sigma_{\mathrm{i=1}}^{\mathrm{m}} \mathrm{x_{ij}^2}}}, (i = 1,2, \dots, m;\ j = 1, 2, \dots, n\,) \quad (6)$$

Step 3: Calculating the weighted normalized decision matrix V=[vij]. Multiply each element rij by the individual weight wj of each criterion:

$$v_{ij} = r_{ij} \cdot w_j \,, (i = 1,2, \dots, m;\ j = 1, 2, \dots, n) \tag{7}$$

Step 4: Determine the ideal positive solution PIS (A+) and the ideal negative solution NIS (A-), where A+ is the maximum benefit and A- is the minimum benefit.

$$A^+ = \{(\max_{i=1}^{n} | j \in I^+|), (\min_{i=1}^{n} | j \in I^-|)\} = [v_1^+, v_2^+, \dots, v_n^+] \,. \tag{8}$$

$$A^- = \{(\min_{i=1}^{n} | j \in I^+|), (\max_{i=1}^{n} | j \in I^-|)\} = [v_1^-, v_2^-, \dots, v_n^-] \,. \tag{9}$$

Where, I+ is the set of benefit-type criteria and I- is the set of cost-type criteria.

Step 3: Calculate the n-dimensional Euclidean distances to the positive ideal solution PIS (A+) and to the negative ideal solution NIS (A-).

$$d_i^+ = \sqrt{\Sigma_{j=1}^{m} \left(v_{ij} - v_j^+\right)^2},\ (i = 1, 2, \dots, n) \tag{10}$$

$$d_i^- = \sqrt{\Sigma_{j=1}^{m} \left(v_{ij} - v_j^-\right)^2},\ (i = 1, 2, \dots, n) \tag{11}$$

Step 5: Calculate the relative closeness c_i^+ for each alternative with respect to the positive ideal solution PIS.

$$c_i^+ = \frac{d_i^-}{d_i^+ + d_i^-}, (0 \le c_i^+ \le 1;\ i = 1,2, \dots,m) \tag{12}$$

Step 6: In the last step of the method, the alternatives are ranked and ordered according to the highest percentage of relative closeness c_i^+ to the ideal solution

EMPIRICAL CASE STUDY

In this study, the classifier models were created and evaluated by regular use of the widget of Orange Data Mining tool, 10-Fold stratified cross validation was performed, which previously classifies the data by classes, and from the same the total data is divided into k folds randomly of the same size, 9 folds are used to train the model and one of the folds is used for testing. The process is repeated 10 times to obtain the performance evaluation result according to each model according to the data set.

RESULTS: EVALUATION RESULTS

From the cross-validation evaluation with the various performance evaluation met-rics to the machine learning classifier models, Table 2. shows the evaluation results of the LR, RF, SVM, kNN, NN, NB models. The performance scores of each dataset (Apache, Safe, Zxing) are averaged.

Table 2. Results of the evaluation

Dataset	LR					RF				
	A	P	R	S	FM	A	P	R	S	FM
Apache	0,505	0,503	0,505	0,496	0,372	0,742	0,743	0,742	0,742	0,742
Safe	0,768	0,766	0,768	0,737	0,767	0,679	0,674	0,679	0,632	0,676
Zxing	0,709	0,670	0,709	0,411	0,660	0,712	0,687	0,712	0,496	0,691
Avg.	**0,661**	**0,646**	**0,661**	**0,548**	**0,600**	**0,711**	**0,701**	**0,711**	**0,623**	**0,703**
Dataset	SVM					kNN				
	A	P	R	S	FM	A	P	R	S	FM
Apache	0,515	0,521	0,515	0,509	0,458	0,716	0,718	0,716	0,715	0,716
Safe	0,732	0,730	0,732	0,698	0,731	0,696	0,694	0,696	0,659	0,695
Zxing	0,363	0,551	0,363	0,595	0,335	0,704	0,691	0,704	0,537	0,696
Avg.	**0,537**	**0,601**	**0,537**	**0,601**	**0,508**	**0,705**	**0,701**	**0,705**	**0,637**	**0,702**
Dataset	NN					NB				
	A	P	R	S	FM	A	P	R	S	FM
Apache	0,670	0,671	0,670	0,669	0,669	0,706	0,707	0,706	0,705	0,706
Safe	0,732	0,741	0,732	0,730	0,735	0,750	0,773	0,750	0,774	0,753

Zxing	0,704	0,689	0,704	0,532	0,695	0,612	0,669	0,612	0,601	0,628
Avg.	**0,702**	**0,700**	**0,702**	**0,644**	**0,700**	**0,689**	**0,716**	**0,689**	**0,693**	**0,696**

At first glance, the RF, NN, NB and kNN models show high performance and are approximately similar. While the SVM and LR models show low performances compared to the other models.

ENTROPY AND TOPSIS RESULTS

Based on the average of the results obtained in the cross-validation evaluation, the following input parameters are used to compare and select the best model using Entropy and TOPSIS. Table 3 shows the decision matrix that is formed by alternatives where LR is A1, RF is A2, SVM is A3, kNN is A4, NN is A5, NB is A6 that will be evaluated based on the criteria where A is C1, P is C2, R is C3, S is C4, FM is C5, with their respective individual weights (wi) associated to the criteria.

Table 3. Decision matrix D = [aij]

	C_1	C_2	C_3	C_4	C_5
w_i	0,219	0,093	0,219	0,127	0,341
A_1	0,661	0,646	0,661	0,548	0,600
A_2	0,711	0,701	0,711	0,623	0,703
A_3	0,537	0,601	0,537	0,601	0,508
A_4	0,705	0,701	0,705	0,637	0,702
A_5	0,702	0,700	0,702	0,644	0,700
A_6	0,689	0,716	0,689	0,693	0,696

Then, by means of (6), the normalized decision matrix is obtained as shown in Table 4.

Table 4. Standardized decision matrix R = [rij]

	C_1	C_2	C_3	C_4	C_5

A_1	0,403	0,389	0,403	0,357	0,374
A_2	0,433	0,422	0,433	0,406	0,438
A_3	0,327	0,361	0,327	0,392	0,316
A_4	0,429	0,422	0,429	0,415	0,437
A_5	0,428	0,421	0,428	0,420	0,436
A_6	0,420	0,431	0,420	0,452	0,433

In order to obtain the weighted normalized decision matrix, the following is used (7). Using (8) and (9) we obtain the ideal solutions A+ and A-. From (10) and (11) the Euclidean distance di+ y di- is calculated. Then with (12) the relative closeness is calculated for each of the alternatives. Finally, the alternatives are ranked based on the Ci+ value, so the best alternative with the value closest to 1 is chosen. The results of the final evaluation and ranking of the alternatives are shown in Table 5

Table 5. Final evaluation and ranking of alternatives

	C_1	C_2	C_3	C_4	C_5	d_i^+	d_i^-	C_i^+	Rank
A_1	0,088	0,036	0,088	0,045	0,127	0,027	0,031	0,532	5
A_2	0,095	0,039	0,095	0,052	0,149	0,006	0,054	0,901	4
A_3	0,072	0,034	0,072	0,050	0,108	0,054	0,004	0,075	6
A_4	0,094	0,039	0,094	0,053	0,149	0,005	0,053	0,916	3
A_5	0,094	0,039	0,094	0,053	0,149	0,005	0,052	0,920	2
A_6	0,092	0,040	0,092	0,057	0,148	0,004	0,051	0,920	1
A^+	0,095	0,040	0,095	0,057	0,149				
A^-	0,072	0,034	0,072	0,045	0,108				

The ranking obtained from all alternatives is A6 > A5 > A4 > A2 > A1 > A3. The best performing alternative is A6 i.e., NB.

CONCLUSION AND FUTURE WORK

In the present work, it was concluded that, in order to shorten testing cycles, reduce effort and increase efficiency in software defect detection, the use of Machine Learning classifier models is very necessary. There are several classifier models, however, there is no one classifier model better than another, because each one performs differently according to the volume and distribution of the data. To overcome this, a hybrid MCDM approach combining entropy and TOPSIS method was proposed to evaluate the performance of the classifier models in order to select the best one. The results obtained show that the hybrid approach is reliable as it is more objective in distributing the weights and more accurate in selecting an efficient alternative. It was determined that the best classifier model is Naïve Bayes compared to other models. For future studies, it is suggested to use other MCDM methods such as VIKOR, ELECTRE or PROMETHEE and combine it with entropy to verify and compare the results obtained by each of them.

ACKNOWLEDGMENTS

This work has been supported by the GIIAR research group and the Universidad Politécnica Salesiana

REFERENCES

C. L. Prabha and N. Shivakumar, “Software Defect Prediction Using Machine Learning Techniques,” Proc. 4th Int. Conf. Trends Electron. Informatics, ICOEI 2020, no. Icoei, pp. 728–733, 2020. [1] C. L. Prabha and N. Shivakumar, “Software Defect Prediction Using Machine Learning Techniques,” Proc. 4th Int. Conf. Trends Electron. Informatics, ICOEI 2020, no. Icoei, pp. 728–733, 2020.

N. Kalaivani, R. Beena, and A. Professor, “Overview of Software Defect Prediction using Machine Learning Algorithms” Int. J. Pure Appl. Math, vol.118, no.20, pp.3863, 2018.

M. Á. Q. Martinez, B. A. M. Tayupanda, S. S. C. Paguay, and L. A. B. Peñafiel, “A Machine Learning Model Comparison and Selection Framework for Software Defect Prediction Using VIKOR,” pp. 890–898, Aug. 2021.

C. Jin, “Software defect prediction model based on distance metric learning,” Soft Comput., vol. 25, no. 1, pp. 447–461, 2021, doi: 10.1007/s00500-020-05159-1.

J. Sun, X. Jing, and X. Dong, “Manifold Learning for Cross-project Software Defect Prediction,” Proc. 2018 5th IEEE Int. Conf. Cloud Comput. Intell. Syst. CCIS 2018, pp. 567–571, 2019, doi: 10.1109/CCIS.2018.8691373.

M. Y. L. Vazquezl, L. A. B. Peñafiel, S. X. S. Muñoz, and M. A. Q. Martinez, “A Framework for Selecting Machine Learning Models Using TOPSIS,” in Advances in Intelligent Systems and Computing, Jul. 2021, vol. 1213 AISC, pp. 119–126.

A. G. C. Pacheco and R. A. Krohling, “Ranking of Classification Algorithms in Terms of Mean–Standard Deviation Using A-TOPSIS,” Ann. Data Sci.,vol.5, no.1, pp.93–110, 2018.

R. Wu, H. Zhang, S. Kim, and S. C. Cheung, “ReLink: Recovering links between bugs and changes,” SIGSOFT/FSE 2011 - Proc. 19th ACM SIGSOFT Symp. Found. Softw. Eng., pp. 15–25, 2011, doi: 10.1145/2025113.2025120.

C. Jin, “Cross-project software defect prediction based on domain adaptation learning and optimization,” Expert Syst. Appl., vol. 171, no. January, p. 114637, 2021.

M. A. Q. Martinez, J. L. M. Redin, E. D. A. Castillo, and L. A. B. Peñafiel, “An Efficient Approach for Selecting QoS-Based Web Service Machine Learning Models Using Topsis,” Lect. Notes Networks Syst., vol. 182, pp. 172–182, Aug. 2020.

C. Pan, M. Lu, and B. Xu, “An empirical study on software defect prediction using codebert model,” Appl. Sci., vol. 11, no. 11, 2021, doi: 10.3390/app11114793.

S. N. A. Saharudin, K. T. Wei, and K. S. Na, “Machine Learning Techniques for Software Bug Prediction:A Systematic Review” J. Comput. Sci., vol.16, no.11, pp.1558–1569, 2020.

E. A. Felix and S. P. Lee, “Predicting the number of defects in a new software version,” PLoS One, vol. 15, no. 3, pp. 1–30, 2020, doi: 10.1371/journal.pone.0229131.

M. A. Q. Martinez, D. T. L. Rugel, C. J. E. Alcivar, and M. Y. L. Vazquez, “A Framework for Selecting Classification Models in the Intruder Detection System Using TOPSIS,” Adv. Intell. Syst. Comput., vol. 1253 AISC, pp. 173–179, Aug. 2020.

J. George, M. Sahu, and H. A. Naqvi, “An entropy-weight based TOPSIS approach for supplier selection,” Irjet, vol. 5, no. 6, pp. 2828–2832, 2018.

R. Singh, D. Kumar, and B. B. Sagar, “Selection of Best Software Methodology Using Entropy and TOPSIS,” ICRITO 2020 - IEEE 8th Int. Conf. Reliab. Infocom Technol. Optim. (Trends Futur. Dir., pp. 81–85, 2020.

M. Nilashi, A. Mardani, H. Liao, H. Ahmadi, A. A. Manaf, and W. Almukadi, “A hybrid method with TOPSIS and machine learning techniques for sustainable development of green hotels considering online reviews,” Sustain., vol. 11, no. 21, 2019.

E. M. Abdelkader, N. Elshaboury, and A. Al-sakkaf, “A Holistic Hybrid Entropy-TOPSIS-based Method for the prioritization of Machine Learning Models in Energy Efficient buildings, 2020 , 7 (9): 61-75 A Holistic Hybrid Entropy-T,” no. September, 2020.