International Journal of Computational Science and Engineering Research
ISSN: 3107 -8605 (Online) , http://www.ijcser.com/
Regular Issue , Vol. 1 , Issue. 4 (October – December) , 2024 , Pages : 51 - 56
Received: 29 May 2024 ; *Accepted:* 28 October 2024 ; *Published:* 30 October 2024
https://doi.org/10.63328/IJCSER-V1RI4P10
Original : Action Research – Compare and Contrast Research Paper

Predictive Analytics for Software Defect Forecastig Based on Machine Learning

B J Priyanka [1]*, S Reddy Bhavana [2], K Pravallika [3], M Sravani [4], P Navya [5]

[1-5] Department of Computer Science and Engineering , Aditya College of Engineering , Madanapalle, India

* *Corresponding Author: B. J. Priyanka; bjpriyanka@gmail.com*

Abstract: The software defect prediction technique yields result that development teams may examine and further contribute to industrial results. It finds all the problematic code portions, helps software developers uncover bugs, and helps them design their testing methods with the help of the model prediction. It is essential to know what percentage of categories yield the accurate forecast for early detection. Moreover, software-defected data sets are supported and at least partially recognized due to their huge dimension. Random forests (RF) and artificial neural networks (ANN) are the machine learning techniques utilized in this research. The forecast for defects is created using historical data. The outcomes showed that the artificial neural network classifier performed better than the random forest classifier.

Keywords: Machine Learning, Artificial Neural Networks, Random Forests, and Software Defect Prediction.

1. Introduction

Software defect prediction, which searches for potential weaknesses in software systems before they manifest in actual software, is a crucial aspect of software engineering. This proactive approach enhances the program's quality and saves a significant amount of money and time on maintenance. Two of the more conventional techniques for finding errors are manual code reviews and testing, both of which are usually insufficient given the increasing complexity of contemporary software systems. Predictive analytics based on machine learning has so emerged as a powerful solution to this problem.

Predictive analytics forecasts the future based on historical data. Models that can estimate the likelihood of errors in newly developed or updated software modules are built using defect data and previous software metrics in the context of software defect prediction. Machine learning techniques, particularly supervised learning algorithms, are perfect for this type of work because of their ability to extract complex patterns and correlations from large datasets. The difficulties in guaranteeing the dependability and quality of contemporary software systems have increased due to their growing complexity and size. Software flaws jeopardize user confidence and system security in addition to causing large financial losses. In order to effectively handle these difficulties, traditional defect detection techniques which frequently depend on manual code reviews and testing are starting to fall short. As a result, in order to foresee and eliminate problems early in the development lifecycle, the software engineering community is increasingly turning to predictive analytics and machine learning techniques. Predictive analytics is a proactive method to software defect control that makes use of statistical models and past data to estimate future events. It is feasible to find patterns and correlations in enormous datasets that point to promise by combining machine learning algorithms.

This study uses multiple machine learning approaches to examine software failure prediction. We look into the effectiveness of several algorithms, including supervised learning techniques like decision trees, support vector machines, and neural networks, as well as unsupervised techniques like clustering. Our study uses a large dataset that contains historical software metrics and defect reports, providing a strong basis for model validation and training. The purpose of this study is to demonstrate how software quality assurance processes could be enhanced by machine learning. By accurately predicting failures, developers may maximize testing resources, concentrate their efforts on high-risk regions, and create software that is more dependable. In this work, two machine learning techniques were used to evaluate the software fault prediction performance. Among the algorithms are Artificial Neural Networks (ANN) and Random Forests (RF). The algorithms have been run on a Jupyter

Jack Sparrow Publishers © 2024, IJCSER , All Rights Reserved
www.jacksparrowpublishers.com

ISSN: 3107 - 8605 (Online) , http://www.ijcser.com/ , Vol. 1 , Issue 4 , 2024 , https://doi.org/10.63328/IJCSER-V1RI4P10

Notebook. The dataset that was used in this study came from a repository that was open to the public. The accuracy of software defect prediction will be evaluated using the various evaluation metrics. We can choose the machine learning algorithm to utilize for predictive analytics in software fault detection based on the outcomes of these tests. Accuracy, recall, f1-score, and precision are among the evaluation metrics. The most suitable machine learning algorithm to use throughout the prediction phase is the one with the highest accuracy. The remaining portion of the paper is organized as follows: Section 2 includes a review of the literature on predictive analytics for machine learning-based software defect detection; Section 3 comprises theory and computation; Section 4 comprises a flowchart and experimental methodology; Section 5 details the outcomes and a discussion of the algorithms' performance; Section 6 offers a conclusion and future directions; and Section 7 includes references.

2. Literature Survey

An example of the empirical literature review of software fault prediction models and methods is given in this section. Below is a synopsis of the in-depth analysis of multiple methodologies that have been offered by different researchers. Finding possible future paths in the defect prediction field which was previously mentioned is the aim of this review.

“Integrated Approach to Software Defect Prediction” by Ebubeogu Amarachukwu Felix, and Sai Peck Lee. An integrated machine learning strategy, based on regression models built utilizing a set of predictor variables, was applied in this research. Regression models use multiple and simple linear regression approaches to estimate the number of flaws in a software product before testing, based on these variables. This technique can estimate the number of defects in a defective program, which sets it apart from previously proposed prediction models for binary defect classification. In binary classification, a program is simply labeled as defective or non-defective without any estimation of the number of defects.

“A Review on Software Defect Prediction Techniques Using Product Metrics” by Jayanthi. R, Lilly Florence and Arti Arya. The volume and complexity of software systems are currently growing at a very quick speed. While there are instances where it enhances performance and yields effective results, there are also instances where it results in increased testing costs, insignificant results, subpar quality, or even untrustworthy items. Software defect prediction is essential to improving software quality and reducing the cost and duration of software testing. Traditionally, software metrics have been used to characterize the complexity and determine how long the programming will take. A thorough study is carried out using software metrics to predict the module's flaws.

Zheng Rong Yang, “A Novel Radial Basis Function Neural Network for Discriminant Analysis”. This research presents a new radial basis function neural network for discriminant analysis. This work, in contrast to many others, focuses on applying the Bayesian technique to utilize the weight structure of neural networks using radial basis functions. It is anticipated that a radial basis function neural network with a thoroughly investigated weight structure will perform better. In this paper, the a priori weight structure of a radial basis function neural network is studied using the Bayesian method because it is typically unknown. This study examines two weight structures: a two-Gaussian structure and a single-Gaussian structure. To estimate the weights, an expectation-maximization learning approach is employed.

Huihua Lu, Bojan Cukic &Mark Culp. “Software Defect Prediction Using Semi-Supervised Learning with Dimension Reduction”. Minimizing non-essential assurance costs and achieving high-quality software products are possible with accurate fault-prone module detection. The availability of software modules with known fault content created in a comparable context is necessary for this kind of quality Modeling. Determining whether a module is defective or not can be costly. The fundamental concept of semi-supervised learning is to augment model training with modules for which the fault information is not available, and learn from a limited number of software modules with known fault content. In this work, we examine how well semi-supervised learning performs in predicting software defects. The method to lower the dimensional complexity of software metrics incorporates multidimensional scaling as a preprocessing technique. The findings of this study demonstrate that the semi-supervised learning algorithm.

Jiajing Wu, Chuan Chen, Zibin Zheng & Michael R. Lyu, “CDS: A Cross–Version Software Defect Prediction Model with Data Selection” (2021). The benefits of cross-version defect prediction (CVDP) over within-project and cross-project defect prediction for real-world applications were covered in this research. Then, they identified two crucial problems that could seriously jeopardize the performance of CVDP models but have rarely been brought up in earlier research. They then carried out exploratory investigations to confirm the existence of these problems and the ways in which they impact CVDP performance. The authors presented a unique cross-version defect prediction model with data selection (CDS), where the defect labels of new and existing files are predicted in different ways, to address these problems and enhance the prediction performance of CVDP. Through rigorous selection and analysis of pertinent data, the data selection

Jack Sparrow Publishers © 2024, IJCSER , All Rights Reserved
www.jacksparrowpublishers.com

ISSN: 3107 - 8605 (Online) , http://www.ijcser.com/ , Vol. 1 , Issue 4 , 2024 , https://doi.org/10.63328/IJCSER-V1RI4P10

process seeks to forecast software problems across various versions.

3. Theory and Calculation

Using machine learning algorithms on historical software development data, predictive analytics for software defect prediction looks for patterns that point to the existence of problems. The main idea is to use historical data to create models that can forecast the occurrence of errors in the future, allowing for proactive measures to improve software quality.

The project's parts include:

- Preprocessing and Data Acquisition
- Data preparation and feature selection
- Model building and training
- Model validation and outcome analysis.

Utilizing the training dataset, train the chosen models. Utilizing the testing dataset and metrics like as accuracy, precision, recall, F1 score, and AUC-ROC, assess the models. An illustration of a calculation Assume we work with a dataset of 10,000 software modules, each of which has 21 features (e.g., number of commits, cyclomatic complexity, and lines of code) to describe it. A binary label in the dataset denotes whether flaws are present (1) or not (0).

Data Splitting:

- Training set: 70% (7,000 modules)
- Testing ser: 30% (3,000 modules)

Interpreting Results:

- Accuracy= (TP+TN)/(TP+TN+FP+FN)
- Precision = TP/(TP+FP)
- Recall = TP/(TP+FN)
- F-Score = (2 Precision Recall)/ (Precision + Recall)

A thorough knowledge of the model's performance is provided by its accuracy, precision, recall, F1 score, and AUC-ROC. High scores for these measures point to a trustworthy model for software defect prediction. Predictive analytics employing machine learning techniques can greatly increase the early detection of software faults by adhering to this theoretical framework and computation procedure. This will improve the quality and reliability of software.

4. Experimental Method

We suggest a productive system that is a part of the conventional machine learning idea and uses artificial neural networks and random forests. The model is trained on a dataset, which reduces execution time and produces effective results. One that usually makes use of a conventional machine algorithm.

Algorithms : Machine learning techniques are utilized to implement predictive analytics for software defect identification through the usage of Random Forest (RF) and Artificial Neural Networks (ANN).

Random Forest: preferred algorithm for machine learning Random Forest is used for both regression and classification tasks. Several decision trees are combined in this ensemble learning technique to produce predictions. A dense network of decision trees is produced by the process, and each one is trained using a random subset of the initial training set's attributes and data. The random sample reduces overfitting and boosts the durability of the model.

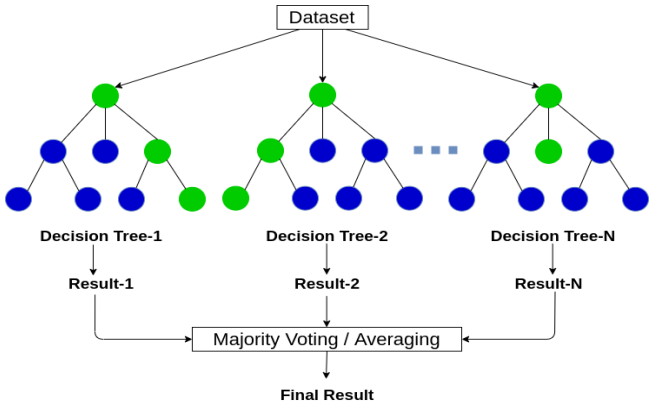

Figure. 1 Random Forest

To produce a prediction, the software aggregates each assumption made by each tree in the forest. In classification tasks, the most common forecast across all trees is selected; in regression tasks, the average of all the forecasts is determined.

Artificial Neural Networks : The connections between distinct neurons and their relative strengths are how our brain processes and stores all of the information that it contains. Neural networks operate on the underlying principle. Neural networks are fundamentally nothing more than a collection of interconnected neurons.

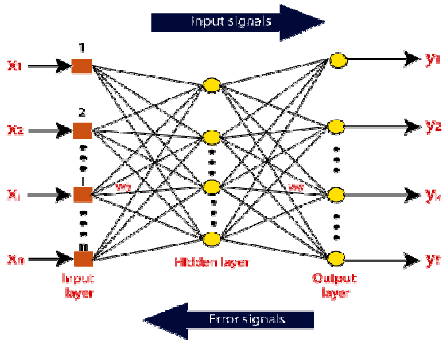

Figure. 2 ANN

Jack Sparrow Publishers © 2024, IJCSER , All Rights Reserved
www.jacksparrowpublishers.com

ISSN: 3107 - 8605 (Online) , http://www.ijcser.com/ , Vol. 1 , Issue 4 , 2024 , https://doi.org/10.63328/IJCSER-V1RI4P10

This introduces a thought-provoking idea: A neural network's structure is unaffected by the task it must do. We need to gain some understanding of the fundamental construction of an ANN (artificial neural network) in order to comprehend that. Three layers of "neurons" can be used to build the most basic ANN. The three layers are the input, hidden, and output layers. Data travels from the input layer to the output layer via the hidden layer, and then output layer.

Flow Chart: Meeting deadlines and finishing all assigned work are crucial. The flowchart is one of many project management tools that are available to assist project managers in keeping track of their tasks and schedule. One of the seven fundamental quality tools used in project management, a flowchart arranges the steps required to complete a task in the most practical order. This kind of tool, often known as process maps, shows a sequence of stages with branching options that represent one or more inputs and convert them to outputs.

Flowcharts have the advantage of providing an overview of the actions involved in a project by mapping the operational details inside the horizontal value chain. This includes decision points, parallel paths, branching loops, and the overall processing sequence. Additionally, this specific tool is highly useful for understanding and assessing the cost of quality for a certain process. This is accomplished by evaluating the projected monetary returns and using the workflow's branching logic.

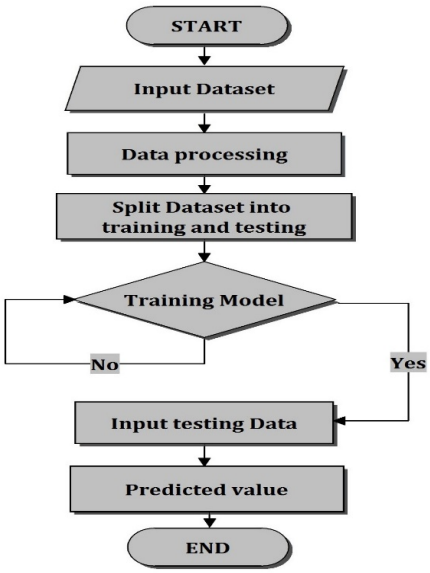

Figure. 3 Flow Chart

5. Result and Discussion

A result is the ultimate outcome of events or actions, either qualitatively or quantitatively stated. An operational analysis is performance analysis, which is a collection of fundamental quantitative relationships between performance quantities.

Table. 1 Metrics

Metrics	Definition
Precision	Precision is defined as the ratio of positive examples to the sum of such actual and false positives
Recall	Recall is defined as the ratio of correct positives to all true negatives and false negatives
F1 Score	A Weight harmonic average such recall and precision is known as the F1. The projected capacity for the model is higher the closer the F1 score value is near 1.0.
Support	The number of instances of a class that truly exist in the dataset constitutes the number of supports . It does not differentiate between kinds, it only improves the performance evaluation process.

Classification Report

	precision	recall	f1-score	support
0	0.70	0.68	0.69	3000
1	0.69	0.71	0.70	3000
accuracy			0.69	6000
macro avg	0.69	0.69	0.69	6000
weighted avg	0.69	0.69	0.69	6000

Figure. 4 Classification Report of Random Forest

The Random Forest algorithm achieved an accuracy of 69% in predicting software defects, showcasing its effectiveness in analyzing and identifying potential issues within codebases. Its ensemble nature, combining multiple decision trees, contributes to its robust performance in handling complex datasets.

Classification Report

	precision	recall	f1-score	support
0	0.87	0.76	0.81	3000
1	0.79	0.88	0.83	3000
accuracy			0.82	6000
macro avg	0.83	0.82	0.82	6000
weighted avg	0.83	0.82	0.82	6000

Figure. 5 Classification Report of ANN

An ANN is giving the best accuracy in this investigation, it means that this specific type of model is performing exceptionally well compared to other models or approaches you've tried. Achieving high accuracy indicates that the ANN is effectively learning patterns in the data and making accurate predictions or classifications.

To provide more context, it would be

Jack Sparrow Publishers © 2024, IJCSER , All Rights Reserved
www.jacksparrowpublishers.com

ISSN: 3107 - 8605 (Online) , http://www.ijcser.com/ , Vol. 1 , Issue 4 , 2024 , https://doi.org/10.63328/IJCSER-V1RI4P10

beneficial to have more information about the particular project, including the dataset you are using, the goal you are attempting to complete (such as regression or classification), the neural network's architecture, and any training or optimization methods you have employed.

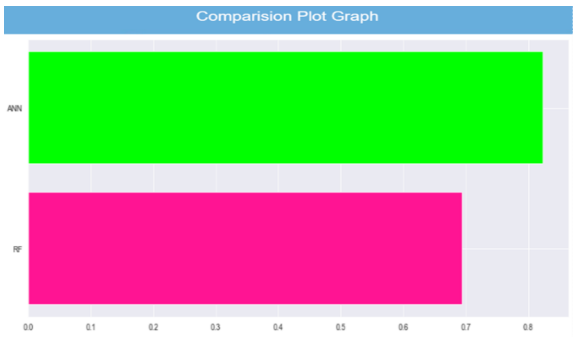

Figure. 6 Comparison graph between RF and ANN

6. Conclusion and Future Scope

Artificial Neural Networks (ANN) and Random Forests (RF) have both been effectively used for software defect prediction. RF is appropriate for a variety of datasets since it frequently offers strong accuracy and robustness against overfitting. An ANN has the ability to detect complex patterns in data and adjust effectively to nonlinear relationships, which may result in good prediction performance. When choose between ANN and RF for software defect prediction, take into account the needs of the task and the particular features of the dataset. The accuracy of the Random Forest (RF) was 69%. The accuracy of the Artificial Neural Network (ANN) was 83%. In terms of accuracy, the ANN performed better than the Random Forest, achieving an accuracy rate of 83% as opposed to 69%.

This shows that the ANN model performed better in accurately predicting software defects and identifying the underlying patterns in the data. When selecting between the two techniques, it is important to take into account additional aspects such processing resources, the interpretability of results, and the particular needs of the software defect prediction task. Overall, even though ANN performed better in this scenario than RF did, the decision between the two should be made after a thorough analysis of all relevant criteria. This presents a viable method to improve the dependability and quality of software. Organizations can proactively identify potential faults and allocate resources for testing and mitigation efforts by utilizing sophisticated machine learning algorithms and historical data.

Teams may identify and fix problems earlier in the development lifecycle using this proactive strategy, which lowers the chance that defects will make it into production and lowers the risks and expenses involved. Furthermore, by providing stakeholders with insightful knowledge about software quality patterns, predictive analytics facilitates well-informed decision-making and ongoing improvement projects. Organizations stand to gain from better software quality, a quicker time to market, and higher customer satisfaction as long as they continue to implement ML-driven defect prediction tactics. Compared to the Random Forest approach, the Artificial Neural Networks (ANN) algorithm yields more accuracy. Potential future projects could involve:

- Exploring advanced ML models like deep learning and ensemble methods.
- Improving feature engineering and automated feature selection.
- Incorporating contextual and temporal analysis for dynamic environments.
- Developing techniques for uncertainty estimation. Enhancing model interpretability and explainability.
- Investigating cross-project transfer learning for resource-constrained settings.
- Implementing real-time prediction and monitoring in live systems.
- Adapting approaches to specific software domains or methodologies.

References

[1]. Felix, E.A. and Lee, S.P., 2017. Integrated approach to software defect prediction. IEEE Access, 5, pp.21524-21547.

[2]. Jayanthi, R. and Florence, L., 2019. Software defect prediction techniques using metrics based on neural network classifiers. Cluster Computing, 22(1), pp.77-88.

[3]. [3] Zheng Rong Yang, “A Novel Radial Basis Function Neural Network for Discriminant Analysis”. IEEE Trans Neural Netw. 2020 May;17(3):604-12. doi: 10.1109/TNN.2006.873282. PMID: 16722166.

[4]. Huihua Lu, Bojan Cukic &Mark Culp. "Software defect prediction using semi-supervised learning with dimension reduction," 2012 Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering, Essen, Germany, 2012, pp. 314-317, doi: 10.1145/2351676.2351734

[5]. Jiajing Wu, Chuan Chen, Zibin Zheng & Michael R. Lyu, “CDS: A Cross–Version Software Defect Prediction Model with Data Selection” IEICE Transactions on Information and Systems, vol. E-95-D, no. 1, pp. 267–270, 2021.

[6]. Abdullah, and Mohammad Zubair Khan. "Software Defect Prediction Using Supervised Machine Learning and Ensemble Techniques: A Comparative Study." *Journal of Software Engineering and Applications* 12.5, 85-100, 2019.

[7]. Iqbal, Ahmed, et al. "Performance Analysis of Machine Learning Techniques on Software Defect Prediction using NASA Datasets." *Int. J. Adv. Comput. Sci. Appl* 10.5, 2019.

Jack Sparrow Publishers © 2024, IJCSER , All Rights Reserved
www.jacksparrowpublishers.com

ISSN: 3107 - 8605 (Online) , http://www.ijcser.com/ , Vol. 1 , Issue 4 , 2024 , https://doi.org/10.63328/IJCSER-V1RI4P10

[8]. Tong, Haonan, et al. "Transfer-Learning Oriented Class Imbalance Learning for Cross-Project Defect Prediction." *arXiv preprint arXiv:1901.08429*, 2019.

[9]. Rahman, Ashiqur. Software Defect Prediction Using Rich Contextualized Language Use Vectors. Diss. 2019.

[10]. Jayanthi, R. and Florence, L., 2019. Software defect prediction techniques using metrics based on neural network classifiers. Cluster Computing, 22(1), pp.77-88.

[11]. Bowes, David, Tracy Hall, and Jean Petrić. "Software defect prediction: do different classifiers find the same defects?" *Software Quality Journal* 26.2, 525-552, 2018.

[12]. Felix, E.A. and Lee, S.P., 2017. Integrated approach to software defect prediction. IEEE Access, 5, pp.21524-21547.

[13]. Xu, Z., Xuan, J., Liu, J., Cui, X.: MICHAC: defect prediction via feature selection based on maximal information coefficient with hierarchical agglomerative clustering. In: 2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER), Suita, pp. 370–381 (2016).

[14]. Wang, T., Zhang, Z., Jing, X., Zhang, L.: Multiple kernel ensemble learning for software defect prediction. Autom. Softw. Eng. 23, 569–590 (2015).

[15]. Y. Ma, G. Luo, X. Zeng, and A. Chen, "Transfer learning for cross-company software defect prediction," Information and Software Technology, vol. 54, no. 3, pp. 248–256, 2012.

[16]. Y. Ma, G. Luo, and H. Chen, "Kernel based asymmetric learning for software defect prediction," IEICE Transactions on Information and Systems, vol. E-95-D, no. 1, pp. 267–270, 2012.

[17]. Y. MA, S. ZHU, Y. CHEN, J. LI, Kernel cca based transfer learning for software defect prediction, IEICE Transactions on Information and Systems 100 (8) (2017) 1903–1906.

[18]. T. Menzies, Z. Milton, B. Turhan, B. Cukic, Y. Jiang, A. Bener, Defect prediction from static code features: Current results, limitations, new approaches, Automated Software Engg. 17 (4) (2010) 375–407.

[19]. Q. Song, Z. Jia, M. Shepperd, S. Ying, J. Liu, A general software defect-proneness prediction framework, IEEE Transactions on Software Engineering 37 (3) (2011) 356– 370

[20]. C. Catal, Software fault prediction: A literature review and current trends, Expert Systems with Applications 38 (4) (2011) 4626 – 4636.

[21]. B. Ghotra, S. McIntosh, A. E. Hassan, Revisiting the impact of classification techniques on the performance of defect pin: Proceedings of the 37th International Conference on Software Engineering - Volume 1, ICSE '15, IEEE Press, Piscataway, NJ, USA, 2015, pp. 789–800

[22]. K. Gao, T. M. Khoshgoftaar, H. Wang, N. Seliya, Choosing software metrics for defect prediction: An investigation on feature selection techniques, Softw. Pract. Exper. 41 (5) (2011) 579–606

[23]. K. Dejaeger, T. Verbraken, B. Baesens, toward comprehensible software fault prediction models using bayesian network classifiers, IEEE Transactions on Software Engineering 39 (2) (2013) 237–257.

[24]. Z. Xu, J. Xuan, J. Liu, X. Cui, Michac: Defect prediction via feature selection based on maximal information coefficient with hierarchical agglomerative clustering, in: IEEE International Conference on Software Analysis, Evolution, and Reengineering, 2016, pp. 370–381.

[25]. D. Ryu, J.-I. Jang, J. Baik, A transfer cost-sensitive boosting approach for cross-project defect prediction, Software Quality Journal 25 (1) (2017) 235–272.

Jack Sparrow Publishers © 2024, IJCSER , All Rights Reserved
www.jacksparrowpublishers.com