International Journal of Internet, Broadcasting and Communication Vol.11 No.2 1-10 (2019)
http://dx.doi.org/10.7236/IJIBC.2019.11.2.1

IJIBC 19-2-1

Investigating the Regression Analysis Results for Classification in Test Case Prioritization: A Replicated Study

Muhammad Hasnain[1], Imran Ghani[2], Muhammad Fermi Pasha[3], Ishrat Hayat Malik[4], Shahzad Malik[5]

[1]*School of IT, Monash University Malaysia*
[2]*Department of Computer Science, Indiana University of Pennsylvania, USA*
[3]*School of IT, Monash University Malaysia*
[4]*School of IT, Monash University Malaysia*
[5]*NUST University Islamabad Pakistan*
Muhammad.malik1@monash.edu, imransaieen@gmail.com,
muhammad.fermipasha@monash.edu, Ishrat.Malik@monash.edu,
shahzad.muet@gmail.com

Abstract

Research classification of software modules was done to validate the approaches proposed for addressing limitations in existing classification approaches. The objective of this study was to replicate the experiments of a recently published research study and re-evaluate its results. The reason to repeat the experiment(s) and re-evaluate the results was to verify the approach to identify the faulty and non-faulty modules applied in the original study for the prioritization of test cases. As a methodology, we conducted this study to re-evaluate the results of the study. The results showed that binary logistic regression analysis remains helpful for researchers for predictions, as it provides an overall prediction of accuracy in percentage. Our study shows a prediction accuracy of 92.9% for the PureMVC Java open source program, while the original study showed an 82% prediction accuracy for the same Java program classes. It is believed by the authors that future research can refine the criteria used to classify classes of web systems written in various programming languages based on the results of this study.

Keywords: *Classification, Test Case Prioritization, Clustering, Replication, Regression Model, and Prediction Accuracy*

1. Introduction.

Classification in software engineering often used to categorize packages into faulty and non-faulty. To categorize packages defect prediction terms are used for software quality classification [1]. In the presence of

Manuscript Received: Feb. 5, 2019 / Revised: Feb. 11, 2019 / Accepted: Feb. 17, 2019
Corresponding Author: imransaieen@gmail.com,
Tel: +1-724-599-8226, Fax: +1-724-357-2724
Department of Computer Science, Indiana University of Pennsylvania, USA

a number software prediction models, it has been left to researchers to determine which prediction models are helpful in varying contexts. To make a clear distinction between learning problems, supervised data is known as classification while unsupervised data is called clustering [2]. For supervised data, an object class is named (label) in advance while for unsupervised data, an object class is not tagged in advance [3]. Therefore, clustering terms are commonly recognized by their unsupervised classifications. Clustering in practice used to find groups that are based on high similarities within data for the same group and low similarities between the data of varying groups. Many clustering approaches have been undertaken in the literature including K-Means by Joseph and Radhamani in [4] and Fuzzy C-Means (FCM) by Lu and Yan in [5]. A new version of the FCM clustering approach was developed, called Kernel Fuzzy C-Means, which focused on overcoming the limitations of existing clustering approaches [6]. Both, classification and clustering have become essential components in data analytics, as well as applications based on machine learning. Besides, these classification and clustering approach, regression analysis as a model has been used for classification of modules which are in more technical sense termed as packages and classes. To classify the packages or classes, the prediction accuracy is used to judge the significance of a classifying approach. Improved prediction accuracy in percentage signifies the classification approach which is used to classify the modules. Increased prediction accuracy of faulty and non-faulty modules helps in ensuring the quality of software. The researchers of this study realized a few limitations of the existing research study for classification of faulty and non-faulty modules. In this paper, the authors extends the primary study of Alsukhni et al. [7], using datasets from open source Java programs and drawing comparisons between the results of two studies within the datasets of two programs. This research study contributes to verifying the classification model used in a recently published research study. The rest of the paper is organized as follows. In Section 2, the authors summarize the background of the original study. Section 3, provides related works. Section 4, presents the experimental setup. Section 5, discusses threats to validity, and the conclusion of this study and ideas for future work are given in Section 6

2. Background to the original study

The Author's goal in the original primary study was to combine the clustering and classifying approaches for the prioritization of test cases. In order to accomplish this objective, they wanted to classify classes of two Java programs into defective and non-defective. To predict defective classes, Alsukhni et al. in [7] used software quality product metrics. They developed a hybrid model that combined classification and clustering of data sets as test cases. They aimed to prioritize test cases for earlier faults detection by using the regression classifiers and k-means clustering. The proposed approach in the above-mentioned study although showed better performance as compared to the existing approaches, however it requires to verify the results. Average percentage of fault detection (APFD) results indicated that testing the default modules of a software was more useful rather than testing the all modules arbitrary. In the original study, datasets of two open sources Java programs named "PureMVC" and "Moss" were used. The authors of the original study used open source Java applications including the 'Metric 1.3.8 Eclipse plug-in' and 'CodeProAnalytix 7.1.0' in the Eclipse IDE to determine metric values and generating test cases, respectively. Logistic regression approach was used to classify defective classes from non-defective classes in two Java programs.

2.1 This Replication

The motivation for carrying out this replication was to further examine whether the use of the same dataset results in the same metrics values, which helps authors verify the regression model for the classification of

Java classes for 'PureMVC', which was used in the original study, and Java program 'Log4j-1.2', which was not used in the original study. The authors believe that it is important to verify results and their interpretation regarding the used regression model. The authors of this paper also wanted to confirm whether or not the regression model used in the original study was accurately producing the same results as the replicated study. The aim of this replicated study is to broaden the results of the original study [7] with the same datasets used in the study as well as additional datasets. If the results are consistent, it may help other researchers in generalizing findings in a wide range of software communities, including software developers and software testers. If results of this replicated study are inconsistent to the original study, this may be cause to reconsider the regression model or any other approach for improving the accuracy of the regression model for Java class classification.

3. Related work

In this section, a brief description of clustering and software metrics related concepts is provided as follows:

3.1 Clustering and Classification Algorithms

Determining cluster centers has become an open issue for researchers as the accurate calculation of cluster centers removes fuzziness and uncertainty related to fuzzy time series. To overcome fuzziness and related uncertainty concerns, Saberi et al. in [8] introduced a fast and precise 'FTS algorithm' for addressing classification and regression problems. In the proposed 'Fast and Efficient Fuzzy Time Series' FEFTS algorithm, several clusters are composed of input data and each cluster has a specific and defined probability density function. The output is calculated from weighted function sums for input variables. To adjust the parameters, 'Least Square Estimation Method' LSEM was applied to six datasets. In order to become familiarized with clustering in software regression testing, Hasan et al. in [9] found that test case clustering from code coverage remained less effective for earlier fault detection in regression testing. This was due to threats in code coverage information that consecutively revealed similar faults. Furthermore, researchers in the lateral study also noted that similar test cases in each cluster of test cases presented the same faults. To overcome issues with the same cluster sharing the same faults, researchers proposed a dissimilarity based TCP approach that incorporates historical failure and similarity data from previous and new application versions. Defects4j datasets were used by researchers to test the proposed technique and the results were compared to other several TCP techniques. Therefore, it was found that dissimilarity based TCP techniques performed better than random, untreated, and similarity-based TCP techniques. Wang et al. in [10] proposed an improved Apriori algorithm for finding association rules. The association rule mines data by combining the algorithm and model and observing vital linking among variables. With these association rules, their proposed algorithm-based model was reconstructed in order to implement regression and classification to improve prediction accuracy. In addition to accuracy, efficiency is a crucial factor in regression and classification. A newly proposed classification and regression approach called 'CRQAR-tree' that was created in [10] and it was mainly composed of 'association classification' and 'rule-based TS fuzzy inference' that employs quantitative associations to enhance prediction accuracy. Although, the improved algorithm and CRQAR-tree approaches are feasible for industrial applications, they show their limitations when used for dynamically changing systems. It is, therefore, required to extend or increment the algorithm by investigating optimized approaches for improving accuracy and efficiency. In an earlier study, Islam and Sakib in [11] proposed 'Package Based Clustering' or PBC to predict software defects. In their proposed clustering, object-oriented classes in Java are grouped into a number of clusters. Cluster findings were based on a working PBC that lists out all object-

oriented classes using text analysis. After reading these classes PBC extracts package information in order to form clusters. Once clusters have been composed, researchers apply linear regression as a prediction model to predict defects and incorporate the eight code metrics. In PBC, class identification is done via considering the extension of files as .java. In the clustering phase, PCB allocates each identified class to a package. PBC as a clustering model was applied to an open source software JEdit 3.2 and PBC performance was compared to the BorderFlow and K-means algorithms. The PBC algorithm outperformed the other two algorithms in terms of percentage performance for predicting defects in JEdit 3.2 software. However, the validation of PBC remains to be addressed in future studies by applying it to more systems with an increased number of software prediction metrics. Joseph and Radhamani in [4] a recently published article preferred clustering over classification regarding test cases. The authors of the same study specifically focused on k-means, fuzzy C-means, and Y-means clustering algorithms. The suitability of K-means algorithms for larger datasets has been not proven; hence the researchers found that fuzzy C-means as an unsupervised clustering algorithm allows the same item to associate with more than one cluster. Therefore, the meaning of C in clustering and fuzzy meaning is unclear or has not been defined. Overall, the performance of the fuzzy k-means algorithm (hybridized by k-means and fuzzy C-means) in terms of computing a large number of variables has been improved compared to other algorithms. Before the above-mentioned fuzzy k-means clustering approach was undertaken, Lu and Yan in [5] proposed an improved 'fuzzy C-means' FCM algorithm that converts clustering problems into mathematical problems. Within the proposal of the improved FCM algorithm, granular computing was combined with FCM to optimize cluster size and sensitivity for data initializing. The following section gives an overview of a few important tools that have been widely used for implementing the classification and clustering approaches.

4. Experimental set up

To perform comparisons of the existing software fault prediction approach proposed in a research study [7], we used datasets freely available on their website as shown in Table 1. In the lateral research article, the authors used open source software including Eclipse, Metrics Plugin 1.3.8, and CodePro AnalyticX. The purpose of using this software is the following.

1. Eclipse provides 'Integrated Development Environment' IDE for the development of Java applications.
2. Metrics Plugin 1.3.8 is widely employed for calculating software attribute values.
3. CodePro AnalytiX is used to generate test cases for imported java programs in Eclipse IDE.

There are a number of classification and clustering approaches that have been widely examined in the literature. One of these classification approaches is Logistic Regression, which was used for classifying fault and non-fault classes by Alsukhni et al. [7] in which researcher proposed the test case prioritization technique. To identify error prone modules in the targeted programs, they used regression analysis to classify classes into faulty and non-faulty. In order to run experiments on two open source java program, they applied software product metrics. To confirm the results obtained in the lateral study and various other studies, we performed complexity and code text related metrics. To evaluate regression analysis with datasets using another program, we selected the Log4j-1.2 an open source java program. Another program used in this study is PureMVC, which was also used in the preliminary study [7].

Table 1. Websites for Open Source Java Programs

Name of application	Website
Log4j-1.2	https://www.apache.org/dyn/closer.lua/logging/log4j/2.10.0/apache-log4j-2.10.0-src.zip
PureMVC	https://github.com/PureMVC/puremvc-java-standard-framework

Table 1 provides information about program name and website addresses from where these java programs can be found. The authors of this paper further classified these java programs into packages and relevant classes as shown in Table 2 and Table 3.

Table 2. Test Cases for PureMVC Program

Application Name	Packages	Classes	Test Cases	Issues
PureMVC	org.puremvc.java.core	3	0	0
	org.puremvc.java.interfaces	11	0	53
	org.puremvc.java.patterns.command	2	1	5
	org.puremvc.java.patterns.facade	1	5	24
	org.puremvc.java.patterns.mediator	1	4	11
	org.puremvc.java.patterns.observer	3	16	31
	org.puremvc.java.patterns.proxy	1	6	11
Total	7	22	32	180

As shown in Table 2, after running the PureMVC program in the CodePro Analytix tool, a total 32 test cases were generated. The number of test cases produced either by running individual packages or running all packages together were the same. However, variance was observed in a number of issues. Table 2 shows that 22 classes from 7 packages resulted in 32 test cases and 180 issues. However, the number of issues was reduced to 135 when running all packages at once. To determine the number of test cases for each package, we used CodePro AnalytiX, an automated tool with various other features such as code analysis, error detection, test generation, and code metrics. As shown in Table 2, no test cases were generated for the two classes. Therefore, classes which resulted in generated test cases were 0, but they can be manually generated. Researchers in the original study of Alsukhni et al. [7] reported manual test case generation for a few classes of the PureMVC program. Similarly, for a number of packages and classes, CodePro AnalytiX cannot produce test cases for its limitations as given in Table 3. The author of this study used the above-mentioned CodePro AnalytiX tool, as the researchers of the original study used it to generate test cases for two open source Java programs.

Table 3. Test Cases for Log4j-1.2 Program

Application Name	Packages	Classes	Test Cases	Issues
Log4j-1.2	org.apache.log4j	14	153	217
	org.apache.log4j.config	7	14	29
	org.apache.log4j.helpers	2	0	0
	org.apache.log4j.layout	1	0	0
	org.apache.log4j.pattern	2	0	0

	org.apache.log4j.spi	9	0	29
	org.apache.log4j.xml	2	33	26
Total	7	37	200	301

Table 3 shows the number of packages, classes, test cases, and issues detected for the Log4j-1.2 program. As shown in Table 3, 7 packages, 37 classes, 200 test cases, and 301 issues pertaining to Java program Log4j-1.2 were used in this study. The aim of using Log4j-1.2 Java program was to observe variance in the results by using another open source Java program. Regarding the test cases shown in Table 3, the authors observed that the CodePro AnalytiX tool could not generate test cases for fourteen classes. The author of this paper reported findings on the limitations of CodePro AnalytiX. However, this limitation is out of the scope of this replication study, as the authors were interested in classifying the classes of open source Java programs.

4.3 Comparison with the Results of the Original Study

This study did not agree with Alsukhni et al. [7] results and their interpretations in some cases. To compare the results obtained in this study with the results of previous study, we believe that binary logistic regression analysis provides a better prediction of Java class classifications. Overall percentage results for the classification tables of both studies were closely related, supporting the results of Alsukhni et al. [7] for the classification of Java classes. One of the main confusing points, which was observed is that how authors of the earlier primary study declared java classes as faulty or non-faulty. In Table 4, the authors of this study showed their factor studied and comparisons with the results of study [7]. The reason for using the 10 factors as given in Table 4 was that the researchers of the original study applied them in their experiment and a few factors like independent and dependent variables were implicitly stated. Therefore, the authors of this study focused on these factors, which were examined in the original study and extended the original study in the context of factors 8-9, which were not discussed in detail in the original study.

Table 4. Comparison of Results

Sr. #	Studied Factors	Alsukhni et al. [7]	Observation
1.	Java Packages	Examined	Confirmed
2.	Java Classes	Examined	Confirmed
3.	Classes Prediction	Examined	Confirmed
4.	Test Cases	Examined	Confirmed
5.	Overall Percentage	Examined (named as Total)	Confirmed
6.	Cyclomatic complexity	Examined	Confirmed
7.	LOC	Examined	Confirmed
8.	Independent Variables	Not Examined	Not Confirmed
9.	Dependent Variables	Not Examined	Not Confirmed
10.	Java Classes Faulty Feature	Not Shown	Not Confirmed

Table 4 shows that the authors of this study failed to find significant differences in the studied factors of two research studies. Therefore the studied factors (1 to 7) and their results are in line with the results of the related study by Alsukhni et al. [7]. However, the authors of this study cannot support the experiments of Alsukhni et al. in terms of the independent and dependent variables used for binary logistic regression analysis in [7].

Researchers in the originally study did not explicitly provide detailed information on the independent and dependent variables they used in their experiments. The results of this study differ from Alsukhni et al. [7] and provide evidence on the use of independent and dependent variables as shown in Table 4. Overall, the majority of the studied factors and the results obtained in this study are in line with the research study [7]. This study was performed on Java programs, similar to the original study where researchers too used Java programs. We tried to focus on the Java programs to see the differences in java programs' size with the same programming language, and module size. We changed a few metrics to see that how results are found to be different from the results of the original study. Both of the original study and this replicated study were shown with the common characteristics as given in Table 4. However, this study presented some of the additional results about independent, dependent variables and java class faulty features. These additional results were missing in the original study. Without these additional results, we were not able to see the variance in classification of faulty and non-faulty classes regarding test cases prioritization. Figure 1 below shows support for the earlier study in the context of the examined factors.

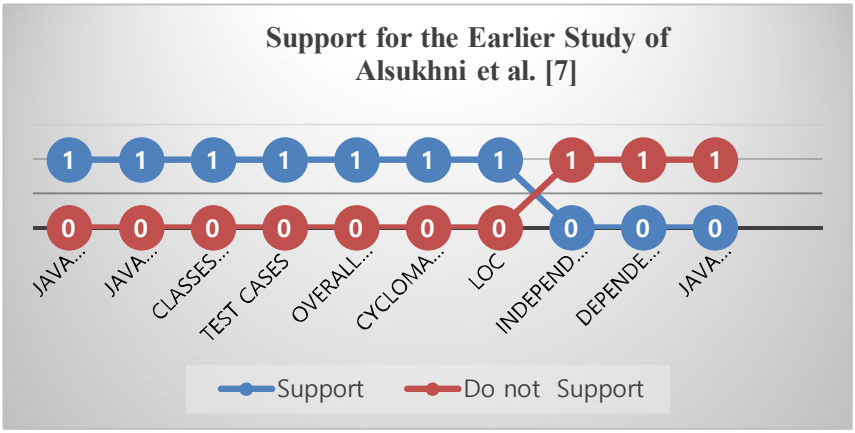

Figure 1. Showing Support for the Earlier Study

Figure 1 illustrates that three studied factors were not supported by results of our study. These studied factors include 'Independent Variables', 'Dependent Variables', and 'Java class faulty features'. Most of the studied factors were examined in both studies and were confirmed by our results. In Table 5, the percentage accuracy results of three open source Java programs are given. Both of the studies are consistent with the majority of studied factors, and a few factors were not concisely evaluated in the original study. It remained as a part of this study to include the missed factors to make this study to complement the original study. This replicated study provided a strong support for the classification of classes regarding the software metrics and straightforwardly letting the researchers to perform test case prioritization.

Table 5. Prediction Accuracy Results of Three Open Source Java Programs

Java Programs	Classification Table Results Constant Not Entered. Overall (%) Accuracy		Predicted Classes (%) Faulty and Non-Faulty			
	Original Study	This Study	Original Study		This Study	
			Faulty	Non-Faulty	Faulty	Non-Faulty
PureMVC	82%	92%	82%	82%	100%	83.3%
Log4j-1.2	x	86%	x	x	87.5%	87.5%
Moss	93.2%	x	0	100	x	x

Table 5 shows the overall percentage accuracy in the prediction of Java classes as faulty and non-faulty. The PureMVC programs results show greater accuracy in this study at 92% compared to the prediction results (82%) of the original study. For individual prediction of faulty and non-faulty classes in both primary studies, the results of this study for faulty and non-faulty Java classes of PureMVC program show a better percentage accuracy of 100% and 83.3%, respectively. In the original study, the percentage accuracy for the prediction of faulty and non-faulty Java classes was reported as 82% and 82%, respectively. For Log4j-1.2 the overall percentage accuracy remained at 86%, and the study of Log4j-1.2 was not undertaken in the original study. This improvement in percent accuracy of faulty and non-faulty classes of Java programs is understandable after performing the binary logistic regression analysis. In Figure 2, a graphical representation of the prediction accuracy results undertaken in the primary studies is given.

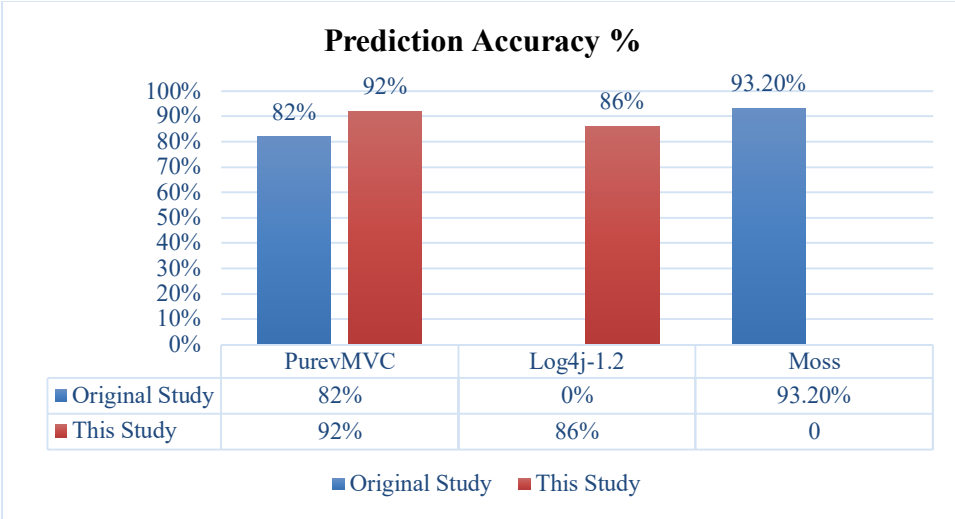

Figure 2. Prediction Accuracy in Two Studies

Figure 2 presents the prediction accuracy in percentage results for two studies regarding PureMVC, Log4j-1.2, and Moss. Although the prediction accuracy percentage for the Moss program has been reported as (93.2%) in the original study, the 92% prediction accuracy for the PureMVC program in this study was far greater than the 82% reported in the original study. In addition, the results obtained for Log4j-1.2 of 86% are also encouraging for researchers in the context of predicting accuracy percentage, as this was better than the PureMVC results in the original study. This confirms that the percentage prediction accuracy of the Java programs may vary from program to program. However, for PureMVC the same program had 22 classes in both studies, showing variations in the percentage prediction accuracy results. Improvements in percentage prediction accuracy results might be a stimulating fact for carrying out research in regression analysis for predicting faults in future research studies. Figure 3 shows a comparison of the original study and this study's results regarding faulty and non-faulty classes for the PureMVC program.

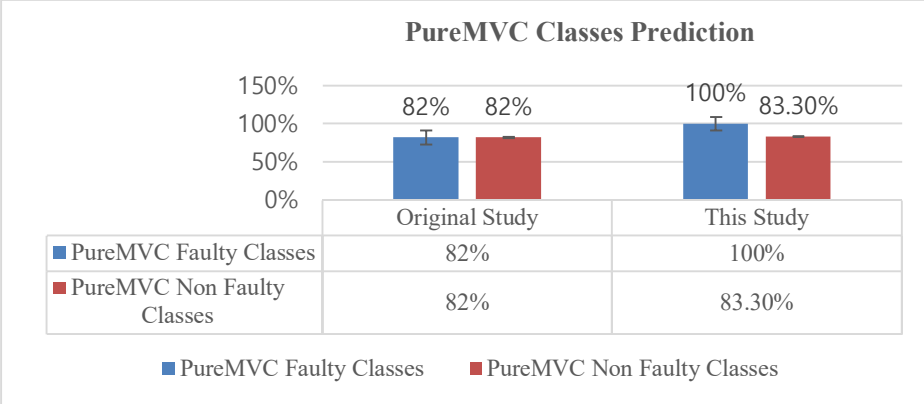

Figure 3. Faulty and Non-Faulty PureMVC Program in Two Studies

Figure 3 shows predictions for the faulty and non-faulty classes of the PureMVC program in the original study as well as this study. In both class predictions, the results of this study show improvements in predicting classes as faulty or non-faulty over the original study. As shown in Figure 3, the prediction of faulty and non-faulty Java classes for the PureMVC program remained at 82% for both cases, it improved to 100% and 83.30% for faulty classes and non-faulty classes, respectively. The authors of this paper showed great interest in replicating the results for the PureMVC and Log4j-1.2 java programs. They used the same tools that were employed in [7] and extended their research in terms of using more statistical tests. We have found that changes in metrics have low impacts on the statistical results which were received using the Metric 1.3.8 Eclipse plug-in' and 'CodeProAnalytix 7.1.0' programs. Binary logistic regression analysis made a little difference between the results of two studies. Results of this study were found significant as series of statistical results provided a clear and concise mean to understand that how faulty and non-faulty classes of Java programs were predicted. Relative improvement in percentage accuracy in prediction might be due to arrangement of metric results, and performing regression analysis on the data collected from metrics results.

5. Threats to Validity

The first external threat to the validity of our experiments may be the use of open source datasets, which may have been changed, slightly impacting the results of our study. The authors of this study tried to collect datasets that used at least one same open source Java program. Data interpretation of the statistical results may vary, which may impact the validity of the results. In addition to above-given validity threats, there may have been fatigue effects that result in decreased motivation. The authors have no means of measuring these fatigue effects. They tried to minimize fatigue effects by using multiple techniques and limiting experiments to only 3 hours per day.

6. Conclusion and Future Work

This paper replicated the study of Alsukhni et al [7]. With regard to the contributions of this study, the authors used datasets from the same java program which were used in the original study. The results obtained in this study verify the binary logistic regression model for the classification of java classes can help in predicting java classes as faulty or non-faulty. It has been found that the binary logistic regression model does not provide a means of identifying java classes as faulty or non-faulty at the individual level. The significant values confirmed that the predicted values were able to examine each category of the dependent variable. However, a legitimate question is whether the model used for identifying faulty and non-faulty java classes in the original

study has assumption based on some statistical results that were not shown in the original study. The replicated study adds some concrete findings regarding the classification of Java classes using software metric values. However, a major research area of classifying individual classes was identified, which can be addressed in future research. Moreover, scalability of regression model can be investigated by using the large scale web systems in future works.

References

[1] Pedrycz, W., Succi, G., & Sillitti, A. (Eds.). Computational intelligence and quantitative software engineering (Vol. 617). Springer. 2016.
DOI: https://doi.org/10.1007/978-3-319-25964-2_1

[2] Celebi, M. E. (Ed.). Partitional clustering algorithms. New York. Springer. 2014.
DOI: https://doi.org/10.1007/978-3-319-09259-1

[3] Saxena, A., Prasad, M., Gupta, A., Bharill, N., Patel, O. P., Tiwari, A….. & Lin, C. T. A review of clustering Techniques and developments. Neurocomputing, 267, 664-681. 2017.
DOI: https://doi.org/10.1016/j.neucom.2017.06.053

[4] Joseph, A. K., & Radhamani, G. Hybrid Test Case Optimization Approach Using Genetic Algorithm With Adaptive Neuro Fuzzy Inference System for Regression Testing. Journal of Testing and Evaluation, 45(6). 2283-2293. 2017.
DOI: https://doi.org/10.1520/JTE20160137

[5] Lu, W. J., & Yan, Z. Z. Improved FCM algorithm based on k-means and granular computing. Journal of Intelligent Systems, 24(2), 215- 222. 2015.
DOI: https://doi.org/10.1515/jisys-2014-0119

[6] Nguyen, D. D., Ngo, L. T., & Pedrycz, W. Towards hybrid clustering approach to data classification: Multiple kernels based interval-valued Fuzzy C-Means algorithms. Fuzzy Sets and Systems, 279, 17-39. 2015.
DOI: https://doi.org/10.1016/j.fss.2015.01.020

[7] Alsukhni, E., Saifan, A. A., & Alawneh, H. A New Data Mining-Based Framework to Test Case Prioritization Using Software Defect Prediction. International Journal of Open Source Software and Processes (IJOSSP), 8(1), 21-41. 2017.
DOI: 10.4018/IJOSSP.2017010102

[8] Saberi, H., Rahai, A., & Hatami, F. A fast and efficient clustering based fuzzy time series algorithm (FEFTS) for regression and classification. Applied Soft Computing, 61, 1088-1097. 2017.
DOI: https://doi.org/10.1016/j.asoc.2017.09.023

[9] Hasan, M. A., Rahman, M. A., & Siddik, M. S. Test Case Prioritization Based on Dissimilarity Clustering Using Historical Data Analysis. In International Conference on Information, Communication and Computing Technology (pp. 269-281). Springer, Singapore. 2017.
DOI: https://doi.org/10.1007/978-981-10-6544-6_25

[10] Wang, L., Li L., Sun, H., & Peng, K. X. A classification and regression algorithm based on quantitative association rule tree. Journal of Intelligent & Fuzzy Systems, 31(3), 1407-1418. 2016.
DOI: 10.3233/IFS-162207

[11] Islam, R., & Sakib, K. A Package Based Clustering for enhancing software defect prediction accuracy. In Computer and Information Technology (ICCIT), 2014 17th International Conference on (pp. 81-86). IEEE. 2014.
DOI: 10.1109/ICCITechn.2014.7073117