INTERNATIONAL JOURNAL OF ADVANCE RESEARCH, IDEAS AND INNOVATIONS IN TECHNOLOGY

ISSN: 2454-132X
Impact factor: 6.078
(Volume 6, Issue 4)
Available online at: www.ijariit.com

Software Defect Prediction by optimizing features weight with a CNN

Asheesh Raju
Alakh Prakash Goyal Shimla University, Shimla, Himachal Pradesh

Anuj Gupta
Alakh Prakash Goyal Shimla University, Shimla, Himachal Pradesh

ABSTRACT

Machine Learning approaches are helpful & have well-tried to be helpful in resolution issues & technical problems that lack data. In most cases, the package domain issues may be characterized as a method of learning that depends on the assorted circumstances and changes of the technical issue being addressed in keeping with the principles of machine learning, a prophetic model is made by exploitation machine learning approaches and classified into defective and non-defective modules. Machine learning techniques facilitate developers to retrieve helpful data when the classification of kinds of technical problems being addressed in an exceedingly specific field. This successively permits them to analyze knowledge from totally different views, which may be used because of the formation base of constructive concepts & varied techniques to handle the technical problems. Machine learning techniques are well-tried to be helpful within the detection of package bugs. during this analysis prediction by Convolution based mostly feature choice and Learning by Random forest. In the proposed approach, the accuracy and precision always improve and it also improves class wise. There is a significant enhancement in defective and non-defective class prediction as the random forest non-linearity features help to improve the selection of effective parameters by bagging approach. In the proposed approach, hybridization of three approaches such as deep learning, machine learning and sampling approach is done which significantly improve overlapping of features and imbalance of class like KC2 dataset.

Keywords— *Machine Learning, CNN, Software Defect Prediction, Random forest*

1. INTRODUCTION

A DEFECT / BUG program is a problem in a software product that does not satisfy a demand for functionality or end-user requirements. In other words, a fault is a coding or logic error which causes a program to defect or generate wrong/ unanticipated outcome.

A system having a significant number of vulnerabilities is called unstable.• Reports that describe program glitches are considered error reports. Bug-finding programs are regarded as error detection devices. The method of bug-finding is called debugging. The deliberate practice of inserting bugs into a software system to approximately check coverage by tracking the identification of those bugs is defined as bugging.

1.1 Software Defect Classification

Software Defects/Bugs are generally classified as per [51]:

(a) Severity / Impact: Fault SEVERITY or Impact is a software fault (bug) designation which indicates the degree of negative effect on software quality.

(b) Probability / Visibility: DEFECT PROBABILITY, also known as Error Visibility or Failure Probability or Failure Visibility, shows the probability that a recipient may find the defect/bug.

- High: reached by all or nearly all feature users
- Medium: encountered by around 50 per cent of function users.
- Low: Found by very few application users

Defect Probability can also be denoted in percentage (%).

(c) Priority / Urgency: Fault PRIORITY, also recognized as Error Priority, shows how critical or urgent a fault is to be repaired. While the Program Tester will originally set preference, the Project/Product Manager typically finalizes it.

(d) Related Dimension of Quality: This involves evaluating the system's accessibility, flexibility, competition, quality, functionality, deployment capability, maintenance, consistency, portability, durability, monitoring, usability as well as protection of the system.

(e) Related Module/Component: Linked Applications/ Devices suggest the program framework or system in which the fault is found. It offers details regarding that whether the component/module is unstable or unsafe. Module/ Component A, B, C

(f) Phase Detected: This indicates the phase in the software development lifecycle where the defect was recognized.

- Unit, Integration, System, Acceptance Testing

(g) Phase Injected: Stage Injected shows the point in which the error was inserted in the software creation lifecycle. In the lifecycle of software creation Process, injection is often

faster than the steps Observed. Only after careful root-cause examination of the problem will the Process Injected be identified.

- Requirements Development
- High Level Design
- Detailed Design
- Coding
- Build/Deployment
- Phase Detected
- Phase Injected

Detecting defects in a Software Project is necessary for the successful implementation & working of the software project. For the reason of project estimation, the below mentioned 4 steps are considered [33]:

- Size estimation of the product development: Lines of Code (LOC) and Feature Points (FP) are available which aid in this form of estimation. However, several other approaches are often used to quantify defects like Use case points (UCP), Story points etc. In this calculation there are other benefits as well as demerits.
- Effort Defect in person-month or person-hour words.
- Failure to plan calendar months.

Project expense Dollar fault, or some other local currency.

1.2 Principles of Defect Prevention

How does a system work in order to avoid faults? The solution falls through a process of avoidance of defects (Figure 1.2). The crucial part of the cycle of fault prevention starts with the design review – converting the consumer expectations into product parameters without making any more errors. Software infrastructure is developed, code analysis as well as checking is performed to evaluate the faults, accompanied by the recording as well as documenting of the faults.

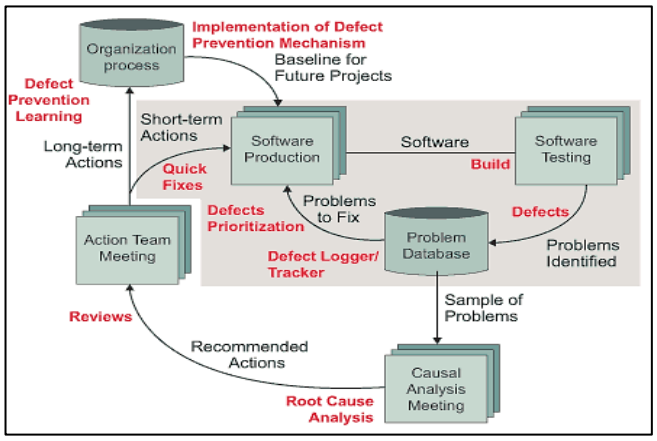

Fig. 1: Defect Prevention Cycle (Source: 1998 IEEE Software Productivity Consortium)

The structures as well as procedures in the gray-coloured framework reflect the handling of defects under much of the software industry's current paradigm–defect identification, tracking/ documentation, and defect evaluation to arrive at fast, short-term solutions. The procedures that make up the essential part of methods for the avoidance of defects are on the white background. The basic step of the technique of defect prevention is to evaluate defects in order to achieve their root causes, to find a swift response as well as preventive intervention. Such prevention strategies are implemented in the company as a model for potential initiatives, with approval and assurances by team leaders. The goal of the technique is to provide the company with a long-term approach and the ability to learn from errors [52]. Many of the techniques of defect prevention practices involve a facilitator. The facilitator may be the group leader of software engineering (wearing another accountability hat) or some aspect of the team. The appointed flaw mitigation coordinator is directly engaged in directing initiatives to eliminate defects, organizing staff and management meetings and coordination, as well as strengthening measures/guidelines for defective reduction.

2. RELATED WORK

Aslı Sar et al. [1] carried out a comprehensive study of CSE literature. The researchers reported 158 studies and 6 secondary studies related to them. They further checked 67 primary studies which carried our standards for quality evaluation. They identified 10 study questions as well as synthesized various methods with respect to each topic included in primary studies. The aim of this analysis is to perform a detailed review of software engineering (CSE) crowdsourcing regarding business models, resources, systems, processes for software creation, but digital economy. Various research teams study crowdsourcing software for coding as well as reviewing activities. Crowdsourcing practices a specific methodology that puts greater focus on project planning, task definition as well as deployment. There is not adequate literature in CSE on strategies to study effort assessment and related cost factors. The nature of the mission as well as its projected length take an important part in predicting it.

Hyunjoo Kim et al. [2] established a model for calculating installation costs via the collection of IFC cost details. This report concentrated on repairing walls of office buildings, and the costs related with the repair. The suggested solution described two key benefits. Next, the substitution details used to equate various situations is immediately retrieved from a BIM file as well as analyzed using IFC to determine a cost estimate. Next, the precision is improved by comparing specific cost-related details, like contractors and suppliers, with the support of CBR in calculating installation costs.

Assia Najm et al. [3] elaborate a comprehensive mapping analysis that categorizes DT articles in line with the following criteria: work methodology, form of input, tools used in conjunction with DT approaches in addition to defining the platforms and patterns for publishing. An automated quest was carried out on five digital repositories to carry out a comprehensive mapping of DT studies, primarily devoted to SDEE conducted in the period 1985-2017. The researchers find 46 studies which are significant. The findings essentially showed that most of the researchers depend on the form of contribution to the methodology.

Przemyslaw Pospieszny et al. [4] Reduces the difference between up-to-date study results as well as operational execution by implementing efficient and realistic machine learning delivery and management strategies, leveraging research findings as well as industry's best practices. This was done by the implementation of ISBSG dataset, smart data planning, an average ensemble of three machine learning algorithms and cross validation. The effort in addition to length calculation models obtained was intended to get a decision-making method for companies designing or integrating information systems.

Ahmed BaniMustafa et al. [5] proposes the design of this forecast utilizing three machine learning methods applied to COCOMO NASA pre-processed test data spanning 93 projects: Naïve Bayes, Logistic Regression and Random Forests. The developed models were cross-validated using five folds as well

as assessed using Classification Accuracy, Precision, Recall, and AUC. The effects of the calculation were then contrasted with that of COCOMO. All the methods used have been effective in obtaining better performance than the COCOMO model as opposed to this model. The best efficiency, however, was obtained using both Naïve Bayes or Random Forests. Due to the fact that in its ROC curve as well as Recall ranking, Naïve Bayes outperformed both the other two methods. Random Forests has a stronger Confusion Index, and scored better in both Identification Accuracy as well as Precision metrics. The findings of this research affirm the relevance of data mining in general, as well as the methodology applied to machine assessment in specific.

Rekha Tripathi et al. [6] present the comparative analysis between traditional techniques and Machine Learning (ML) methods. Findings show that ML approaches have a more reliable estimate of effort relative to conventional methods of estimating effort. In this article, the contrasts of various Machine learning methods are performed to research whether the ML approach is more effective, and in which scenario.

Ashu Bansal et al. [7] stresses the production of a fuzzy multi-criteria-based approach to decision-making by combining Fuzzy Set Theory as well as Weighted Distance Dependent Approximation. To illustrate the accuracy of the suggested technique, framework testing is also performed by comparison with current methodologies. Apart from this, sensitivity review is also conducted to test the criticality of the criteria collection.

Munialo, et al. [8] exhaustively study current software commitment calculation approaches by developing calculation methods tailored to modernise app creation techniques.

Deepika Badampudiet al. [9] Identify considerations that could affect the decision in the literature to select between specific component roots and decision-making approaches (for example, optimization). A systematic review research was performed on peer-reviewed literature. The study conducted a minimum of 24 main trials. The sources of the part were contrasted primarily in-house vs. COTS and COTS vs. OSS. They established 11 factors which affect or influence the decision to choose the origin of a variable**.** When evaluating the origin of the variable, little information existed about the relative influence of a variable origin on the element. Models of optimisation are the methodology most frequently discussed in the solutions.

Tassio Vale et al. [10] investigate the modern CBSE area by a thorough analysis of the literature. To this end, 1231 studies were reviewed that range from 1984 to 2012. Using the available data, this paper discusses five dimensions of CBSE: key goals, study subjects, fields of use, strength of analysis as well as techniques of applied science. The key priorities defined were to maximize efficiency, to save money or boost quality. The technology areas that are often discussed are homogeneously split into commercial-off -the-shelf (COTS), centralized and embedded systems.

Ye Yang et al. [11] presents a conceptual design with a modern pedagogical approach utilizing LEGOs for teaching principles as well as techniques for device calculation as well as measurement. Two case study sessions test the framework: one on seasoned part-time business graduates, and one on novice on-campus graduates. Results from both sessions suggest a good effect on learning for the students.

Sathya, R. et al. [12] recognizes key factors that in effect propose approaches to increase the quality and usability of apps. The paper also illustrates how the different methods of defect prediction are applied, contributing to a decreased severity of faults.

3. THE PROPOSED METHOD

STEP 1: Select data from the promise dataset and divide it into features and labels.
STEP 2: Features are convoluted by convolution layers and mapped by using two activation functions; namely **sigmoid** and **TANH**, because different efficient values come together as a result.
STEP 3: After activation, function is mapped by max polling, then merged in matrix A and labelled in Matrix B at last.
STEP 4: Upon labelling, apply sigmoid function which finally gives us the abstract features.
STEP 5: Features are learned by decision tree and make no overlapped forest.
STEP 6: Out of the forest, find useful trees using the boosting approach, then make the final model and analyse the different parameters.

3.1 Methodology

3.1.1 Convolution with Random forest

STEP 1: In the first step, extract or parse the features of the promise dataset and concentrated with the feature. Both the features are provided with the software's domain-based information, along with the complexity of the software.

STEP 2: After extraction of the features, the label of aging is provided but not the aging. So, the proposed approach initially finds the software module and predicts whether the module is reused or not. If it is predicted, then it is classified according to feature aging but not aging.

STEP 3: Extracted features are convoluted and then the mapping is done by using sigmoid and TANH activation function. These functions not only map the non-linearity of features but also map the bigger value to the abstract value.

STEP 4: The proposed approach uses two types of convolution: local and global-local pooling. This in turn improves the local optimization of features and the global features improve the overall efficiency of the features.

STEP 5: After extraction of the features, reusability-based regression is applied using the following architecture.

$$\begin{aligned} Pr(\mathbf{y}|\mathbf{F}, \Psi) &= g(\mathbf{Z}_{out}\mathbf{W}_{out} + \mathbf{b}_{out}) \\ \mathbf{Z}_{out} &= \sigma(\mathbf{Z}_l\mathbf{W}_l + \mathbf{b}_l) \\ &\cdots \\ \mathbf{Z}_{k+1} &= \sigma(\mathbf{Z}_k\mathbf{W}_k + \mathbf{b}_k) \\ &\cdots \\ \mathbf{Z}_1 &= \sigma(\mathbf{F}\mathbf{W}_{in} + \mathbf{b}_{in}), \end{aligned}$$

STEP 6: After reusability, prediction selects the decision tree by:

$$\mathcal{F}(\Theta) = \{\mathcal{T}_m(\Theta_m)\}, \quad m = 1, \ldots, M,$$

STEP 7: Random forest generates a large number of decision trees and selects an effective tree out of the bulk number of decision trees by the following equation:

$$f(\mathbf{x}_i; \Theta) = (T_1(\mathbf{x}_i; \Theta_1), \ldots, T_M(\mathbf{x}_i; \Theta_M))^T$$

In the above equation, mapping of trees is done and prediction is done depending upon convolution aging classification.

STEP 8: Analyze the predicted model by precision, recall, and accuracy Convolutional Neural Networks (CNN), were first introduced by Yann LeCun's in 1998 for Optical Character Recognition (OCR), where they have shown impressive performance on character recognition. CNN is not just used for image related tasks, they are also commonly used for signals and language recognition, audio spectrograms, video, and volumetric images.

4. RESULT ANALYSIS

4.1 Result Analysis

This part includes the details of the experiment based on different classifiers as represented below:

Table 1: Analysis of the different dataset on the existing and proposed approach

Dataset	Accuracy (CNN)	Accuracy (Ensemble)	Precision (CNN)	Precision (Ensemble)	Recall (CNN)	Recall (Ensemble)
CM1	96.34	94.043 33333	96.45	95.283 33333	95.5291 6667	94.27916 667
JM1	97.45	93.746 66667	98.45	93.913 33333	95.89	94.64
KC1	98.34	93.23	97.67	92.34	97.67	93.34
KC2	94.34	90.45	94.56	89.12	96.34	98.56
PC1	98.56	96.56	99.12	95.12	98.99	98.78

Table 1 analyses the proposed approach and the existing approach performance on different datasets using comparison metrics. The results of the experiment are shown in table 5.1 and the graphical representation is presented via figure 2, 3 and 4 based on accuracy, precision and recall comparison, respectively. The comparison of the results is shown in two aspects. The first aspect is different dataset and the other aspect is the proposed and the existing approach. In the first comparison, accuracy of different dataset varies from 94% - 98% in the proposed approach and the existing approach varies from 90% - 96%.

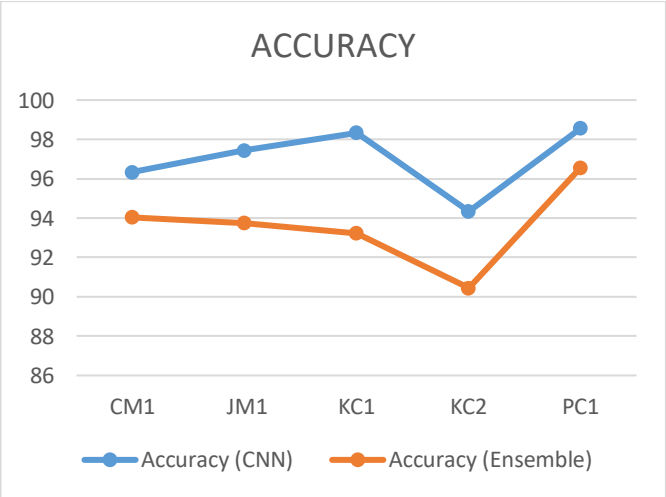

Fig. 2: Accuracy analysis of the different dataset on the existing and proposed approach

In figure 2, the analysis of accuracy is done and it shows the accuracy pattern as same in the proposed and existing approach. In figure 5.1, higher accuracy in KC1 is obtained and minimum accuracy is obtained in KC2. On comparing with the proposed and existing approach, CNN based proposed approach improves accuracy significantly. In table 1, another parameter is precision ranging from 94% - 99% in the proposed approach and the existing approach ranges from 89% - 95%. Figure 3 shows the comparison of precision.

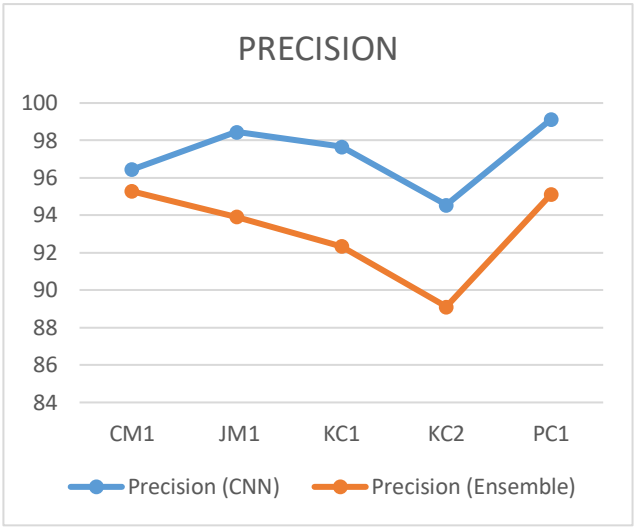

Fig. 3: Precision analysis of the different dataset on the existing and proposed approach

In figure 3, the precision analysis of different PROMISE datasets is done. The graph shows the precision pattern which is not the same as in the case of accuracy. In precision cases, increased precision can be observed in the two datasets i.e. JM1 and KC1. But in this case, KC2 reduces. Whereas, in the case of accuracy, increment & reduction occur only for a single dataset i.e. KC1 and KC2. But in fig 3, the precision of JM1 and KC1 is an approximate value. Similarly, precision and its value are significantly high as compared to the existing approach based on PSO. The last section of this chapter analyses the reason for the performance-based increment of the proposed approach. Table 1 also analyses the recall parameter which varies from 95% - 98% in the proposed approach and 93%-98% in the existing approach. In fig 4, the analysis of the recall pattern comes after the experiment and also the graphical representation of the proposed and existing approach is done.

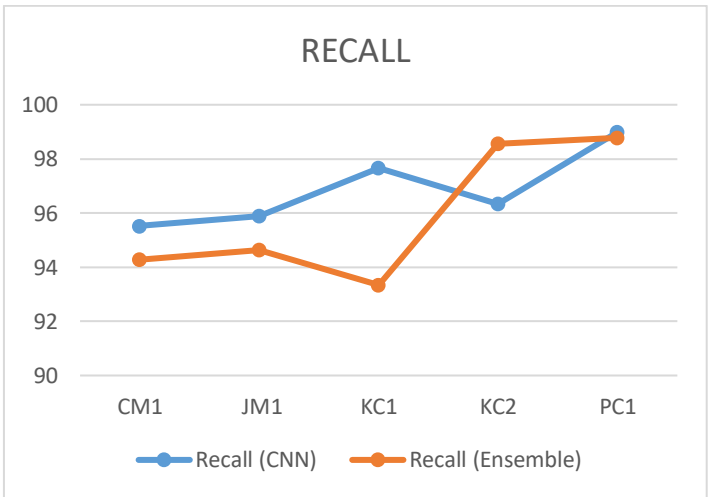

Fig. 4: Recall analysis of the different dataset on the existing and proposed approach

Fig 4 and fig 5 show the recall and comparison analysis respectively. But in fig 4, the analysis represents recall not always but shows signs in the proposed approach in case of KC2 dataset. The average performance of all five datasets recalls

improves effectively in the proposed approach as compared to the existing approach.

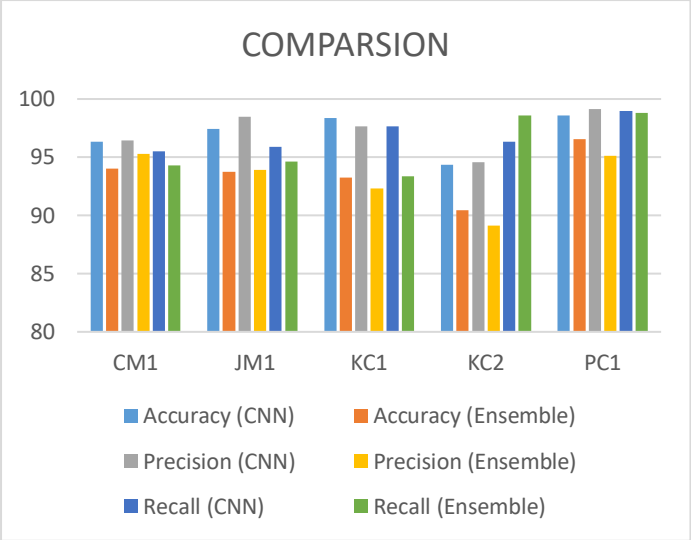

Fig. 5: Comparative Analysis of different dataset on the existing and proposed approach

5. REFERENCES

[1] Sarı, Aslı, Ayşe Tosun, and GülfemIşıklarAlptekin. "A systematic literature review on crowdsourcing in software engineering." Journal of Systems and Software 153 (2019): 200-219.

[2] Kim, Hyunjoo, and Jonghyeob Kim. "A Case-Based Reasoning Model for Retrieving Window Replacement Costs through Industry Foundation Class." Applied Sciences 9, no. 22 (2019): 4728.

[3] Najm, Assia, AbdelaliZakrani, and Abdelaziz Marzak. "Decision Trees Based Software Development Effort Estimation: A Systematic Mapping Study." In 2019 International Conference of Computer Science and Renewable Energies (ICCSRE), pp. 1-6. IEEE, 2019.

[4] Pospieszny, Przemyslaw, Beata Czarnacka-Chrobot, and Andrzej Kobylinski. "An effective approach for software project effort and duration estimation with machine learning algorithms." Journal of Systems and Software 137 (2018): 184-196.

[5] BaniMustafa, Ahmed. "Predicting software effort estimation using machine learning techniques." In 2018 8th International Conference on Computer Science and Information Technology (CSIT), pp. 249-256. IEEE, 2018.

[6] Tripathi, Rekha, and P. K. Rai. "Machine Learning Methods of Effort Estimation and It's Performance Evaluation Criteria." International Journal of Computer Science and Mobile Computing 6, no. 1 (2017): 61-67.

[7] Bansal, A., B. Kumar, and R. Garg. "Multi-criteria decision-making approach for the selection of software effort estimation model." Management Science Letters 7, no. 6 (2017): 285-296.

[8] Munialo, Samson Wanjala, and Geoffrey MuchiriMuketha. "A review ofagile software effort estimation methods." (2016).

[9] Badampudi, Deepika, ClaesWohlin, and Kai Petersen. "Software component decision-making: In-house, OSS, COTS or outsourcing-A systematic literature review." Journal of Systems and Software 121 (2016): 105-124.

[10] Vale, Tassio, Ivica Crnkovic, Eduardo Santana De Almeida, Paulo Anselmo Da Mota Silveira Neto, YguaratãCerqueira Cavalcanti, and Silvio Romero de LemosMeira. "Twenty-eight years of component-based software engineering." Journal of Systems and Software 111 (2016): 128-148.

[11] Yang, Ye, and Linda Laird. "Teaching software estimation through LEGOS." In 2016 IEEE 29th International Conference on Software Engineering Education and Training (CSEET), pp. 56-65. IEEE, 2016.

[12] Sathya, R., and P. Sudhakar. "Improve Software Quality using Defect Prediction Models." International Journal of Engineering and Management Research (IJEMR) 6, no. 6 (2016): 24-29.

[13] Agrawal, Vidisha, and Vishal Shrivastava. "Performance evaluation of software development effort estimation using neuro-fuzzy model." Int. J. Emerg. Res. Manag. Technol 4 (2015): 193-199.

[14] Fehlmann, Thomas. "4.4 When to Use COSMIC FFP? When to Use IFPUG FPA? A Six Sigma View." COSMIC Function Points: Theory and Advanced Practices (2016): 260.

[15] Sarro, Federica, Alessio Petrozziello, and Mark Harman. " Multi-objective software effort estimation." In 2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE), pp. 619-630. IEEE, 2016.

