INTERNATIONAL JOURNAL OF ADVANCE RESEARCH, IDEAS AND INNOVATIONS IN TECHNOLOGY

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

Review on software defect prediction with role of Machine Learning and Feature Selection

Megha Saloni
Yamuna Group of Institutions Engineering and Technology, Yamuna Nagar, Haryana

Sucheta
Yamuna Group of Institutions Engineering and Technology, Yamuna Nagar, Haryana

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

Keywords: *Software, Defect, Estimation, Features, Machine Learning*

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.

h)
- 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]:

1.2 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.

1.3 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.

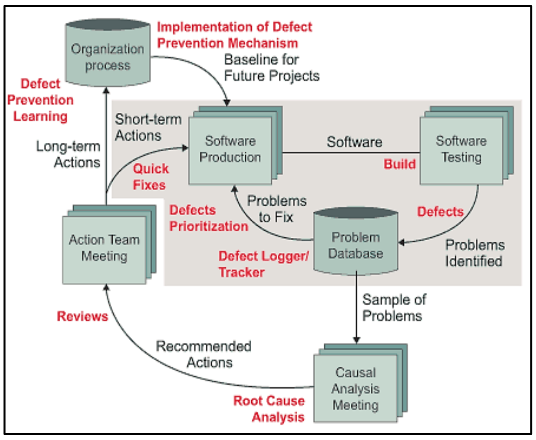

Fig. 1: Defect Prevention Cycle (Source: 1998 IEEE Software Productivity Consortium)

The structures as well as procedures in the gray-colored 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

An accurate measurement of the program scale is the first step in obtaining an efficient project calculation. In comparison to the structured definition of a project specification, various approaches may be applied to assess the extent of the size of fault in the project/product/ program. Different sources of the data available for cost assessments include consumer configuration requirements, demands for suggestions as well as a device or program design definition. Specific information can be given with the aid of layout records in the case that a fault arises in a project in its later lifecycle phases [23]. Two approaches have been defined below for the purposes of product size estimate:

a. **By analogy**: It is simple to include all the necessary calculations for the intent of constructing a new project because one has previously worked with specific styles of projects. It is most likely parallel to the earlier one that seeks to provide the project's overall cost due to its scale. The overall scale amounts to a specific portion in the identical recent project. The measurements of all smaller components shall be used when calculating the scale of a new project. A skilled estimator may make stronger predictions of the scale with the help of analogy. This technique can be proven to be efficient only when the values are of precise size and that of the prior projects obtainable. Besides this, the novel project is similar to the earlier one.
b. **Counting product features along with an algorithmic approach:** Several subsystems, methods/functions are incorporated in the "Package Functions" macro-level.

An integral aspect of Computer Project Planning is calculating the scale of the project. It lets the project manager to foresee better commitment. Specific methods are used for calculating project scale. Many of these are [53]:

Lines of Code (LOC): LOC counts the cumulative number of characters of source code inside a project, as the name implies. The LOC Groups are:

- KLOC- Thousand lines of code
- NLOC- Non comment lines of code
- KDSI- Thousands of delivered source instruction

The scale is determined by matching it with the same kind of current structures. Experts to estimate the appropriate size of various software components, and instead combine them to have the final size use this.

Benefits:

- Universally accepted as well as utilized in a lot of models like COCOMO.
- Estimation is closer to the developer’s perspective.
- Simple to use.

Drawbacks:

- Various programming languages are made of separate sections.
- There is no accepted market standard for this methodology.
- In initial stages of the project, it is impossible to measure the scale using this methodology.

(ii) Number of entities in ER diagram: A static image of the project is represented via an ER model. It defines the individuals and their interactions with each other. A number of individuals may be used in an ER model to calculate project size estimation. The number of individuals in a project relies on the project scale. This is due to the fact that as more organizations require more classes/structures, which will ultimately result in more coding.

Advantages:

- Assessment of the scale will be rendered at initial planning phases.

• Number of individuals is irrespective of the techniques used in programming.

Disadvantages:
• There is no set standard. Many organizations allocate the scale to a proposal than some do.
• Like FPA, this model is less commonly utilized in cost estimation. Therefore, it has to be translated to LOC.

(iii) Total number of processes in detailed data flow diagram: Data Flow Diagram (DFD) reflects a machine interactive view. The layout represents the key program processes/functions involved and data transferred through them. Usage of number of programs in DFD is to estimate the size of the program. Existing similar-type systems are now being researched and then used to approximate the process complexity. The final estimate of the size is given by total of the estimated size of growing phase.

Advantages:
• The programming language is unique.
• Each significant method could be broken down into lesser processes. This will improve the precision of estimation.

Disadvantages:
• It takes extra time and effort to research similar types of processes to measure the scale.
• The construction of DFD is not required for all software projects.

(iv) Function Point Analysis: In this process, FPC is used to identify the amount and form of functions provided by the program. The measures in the study of the feature points are:

(a) Count the number of functions of each proposed type: Find the number of roles that correspond to the following types:
• Internal Data: Abstract information contained inside the program. There are no log files in here.
• External Inquiries: This contributes to device data recovery but will not alter the program.
• External Inputs: Device data entry features.
• External outputs: System-exit application features.
• Remote device Folders: These are functional data that are used by our framework for certain programs.

(b) Compute the Unadjusted Function Points (UFP): Categories dependent on their sophistication, each of the five feature types is either basic, medium or complicated. Multiply the weighting factor of every feature form, as well as find the weighted total. The weighting factors are as described for each form, focused on their difficulty

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 takes 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.

AssiaNajm 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.

PrzemyslawPospieszny 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 Valeet 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.

Vidisha Agrawal et al. [13] the projected time for the Neuro fuzzy model generated for three membership functions is contrasted with current versions of the neural network. Compared with neural network simulations, the Neuro fuzzy construct for Gaussian, triangular and trapezoidal membership function is used. Lopez Martin dataset was used for this analysis with 41 units. Based on five separate parameters, the researchers contrasted the three separate membership structure models to current neural network models. Such models include Mean Magnitude Relative Error (MMRE), Forecast (Pred), as well as Root Mean Squared Error (RMSE). Eventually, it is found from the contrast that the model Neuro Fuzzy uses the feature of Trapezoidal membership and thus, provides better outcomes than all other versions.

Federica Sarro et al. [15] introduce a bi-objective commitment evaluation method incorporating confidence interval analysis as well as the Mean Absolute Error assessment. The researchers are assessing the suggested algorithm on three separate alternate models, reference comparators as well as existing state-of - the-art initiative estimators extended to five Pledge registry real-world datasets, affecting a total of 724 specific software ventures. The findings show that the suggested method surpasses the standard, modern as well as all three alternate formulations in all five datasets, statistically substantially (p <; 0.001) even with broad impact size (A 12 al 0.9).

MirosławOchodek et al. [16] examine how the calculation processes of FPA (Function Point Analysis) and SNAP (Software Non-Functional Evaluation Process) apply to each other, as well as offer some early insights into the use of SNAP to calculate the non-functional device scale. The study findings indicate that SNAP will help to alleviate certain well-known FPA process deficiencies. We have also found several possible issues associated with implementing SNAP in a price-per-size-unit pricing environment, though.

Meenakshi Saroha_et al. [17] provide discussion of different techniques used to estimate effort; however, the primary emphasis is on the resources as well as mechanisms built to estimate effort depends on the Use Case Point (UCP) model. Such tools have additional functions such as including further aspects that may influence the execution of a project as well as the potential to provide a better estimation than current ones.

Wei Lin Du et al. [18] uses an adaptive hybrid model that integrates a neural network model combined with a fuzzy model to enhance the precision of computational cost estimation. The efficiency of the proposal model is measured using reported project as well as industrial data to plan and execute assessments. Results revealed that, it depends on the Mean Magnitude of Relative Error (MMRE) parameter. The suggested model showed the potential to increase the estimation accuracy by 18 per cent.

Lalit V. Patil et al. [19] proposed a hybrid solution, consisting of the Functional Link Artificial Neural Network (FLANN) as well as the teaching algorithm COCOMO-II. FLANN decreases the difficulty of multilayer neural network computations. It has no secret layer, and has the potential to learn quickly.

Leena Nerkar et al. [20] provides a broad description of the algorithmic as well as non-algorithmic machine cost estimating methods/techniques such as the COCOMO concept, Putnam Concept, Model focused on Function-Point, professional decision, comparison prediction, Parkinson's Rule and pricing to gain. Each methodology for estimation has its own pros as well as cons. It also offers a description of Hybrid Concept. This paper's primary aim is to clarify all current electronic cost estimating strategies.

FerruhYiğitet al. [21] using IG (information gain) as well as PSO (particle swarm optimization) algorithms based on a new feature selection method, text classification phase was conducted. The studies used Reuters 21.578, and Classic3 corpus. The origins of the words in the corpus texts have been taken as characteristics. Processes of discovery as well as categorization of features were conducted using IG and PSO algorithms with k-Nearest Neighbors algorithm (K-NN) and Naive Bayes classification. The suggested efficiency of the device was measured using parameters CA (Classification Accuracy), Speed, Remember, as well as F-measure.

M. Madheswaran et al. [22] concerned with developing computational effort calculation model by focusing on artificial neural networks. The program is explicitly built to increase the network performance which correlates to the COCOMO program. In this post, it is proposed that multi-layer feed forward neural network to match the model, as well as its parameters, in order to estimate efforts in app growth. The network is fitted with learning algorithm for spreading back by iteratively checking a series of training tests as well as comparing the network's prediction with the actual effort. The COCOMO dataset is used to train and test the network, and it was observed that the neural network model indicated improvement in the accuracy of the estimation of model.

Gabriela Czibula et al. [23] suggest a novel type of grouping focused on the mining laws for relational associations. Relational connection rules are an extension of ordinal association rules, that are a special form of organization rules that define numerical ordering between attributes frequently occurring over a dataset; our classifier is focused on discovering the laws of relational interaction for determining how well a program module is faulty or not. It offers an initial assessment of the new paradigm on the NASA open source repositories, and a review to related current methods.

Virendra Prabhakar et al. [24] use Artificial Neural Network (ANN) and Help Vector Machine (SVM) to estimate device effort utilizing China dataset. The output indices Sum-Square-Error (SSE), Root-Mean-Square-Error (RMSE), Mean-Square-Error (MSE), Mean- Relative Absolute-Error (RAE), Relative-Root-Square-Error (RRSE), Correlation Coefficient (CC), Mean-Absolute-Errror (MAE), and PRED (25). These were used to evaluate the findings attained from those 2 approaches.

Ali Bou Nassifet al. [25] planned a new log-linear regression model which was dependent on the Use Case Point Model (UCP), to measure the Use Case Diagram program effort. A fuzzy logic methodology is used in the regression model to calibrate the efficiency component. In addition, a neural network model based on multilayer perceptron (MLP) was built to predict software effort dependent on the project size as well as team productivity. Experiments suggest that the solution suggested outperformed the initial UCP model.

2.1 Inferences drawn from literature review:

S no.	Author's Name	Method Used	Paper Title	Application Domain	Inferences
1.	**Przemyslaw Pospieszny et al.**	ISBSG dataset, smart data preparation, an ensemble averaging of three machine learning algorithms (Support Vector Machines, Neural Networks and Generalized Linear Models) and cross validation	*An efficient method for software project effort as well as duration estimation with machine learning algorithms*	Software effort and duration estimation	The commitment and length calculation models obtained was supposed to provide a decision making method for designing organizations or integrating information systems.
2.	***Vidisha Agrawal et al.***	Membership function models (Gaussian MF, Triangular MF and Trapezoidal MF)	*Performance evaluation of software development effort estimation using neuro-fuzzy model*	Neuro fuzzy model for estimating software time	Neuro Fuzzy model of Trapezoidal membership feature performs more than every other configuration.
3.	***Federica Sarro et al.***	Real-world datasets from the PROMISE repository, involving 724 different software projects in total	*Multi-objective software effort estimation*	Bi-objective effort estimation algorithm	The planned algorithm exceeds the standard, modernised well as all the three substitute formulations, which are statistically relevant ($p <; 0.001$) and with large effect size ($A_{12} \geq 0.9$) over all five datasets
4.	***MirosławOchodek et al.***	FPA, SNAP	*Functional as well as Non-functional Size Measurement with IFPUG FPA and SNAP—Case Study*	Measure the non-functional size of applications .	The study findings indicate that SNAP will help to alleviate certain well- FPA process limitations.
5.	***Dan Ingold et al.***		*A model for estimating agile project process with plan acceleration*	Constructive Rapid Application Development Model (CORADMO)	CORADMO attempts to measure the influence of primary program factors and thereby helps managers to predict the relative timetable arising from the variance of certain parameters.

6.	***Mamoona Humayun et al.***	Global software development (GSD), artificial intelligence (AI), machine learning (ML)	*Estimating Effort in Global Software Development Projects Using Machine Learning Tec hniques*	Effort estimation	A qualitative study is performed between conventional approaches for calculating commitment and ML approaches. Results indicate that ML methods provide us a more reliable measurement of effort relative to conventional methods of estimating effort.
7.	***D. Asir Antony Gnana Singh et al.***	Feature subset selection and feature-ranking methods	*An empirical study on dimensionality reduction as well as improvement of classification accuracy using feature subset selection in addition to ranking*	Dimensionality reduction	The techniques employed increase the classifier's prediction precision, reduce the incorrect prediction factor, as well as minimize the cost of time and space to construct the statistical model.
8.	**Martin Shepperd *et al.***	An unbiased statistic, Standardized Accuracy, Random 'predictions'	*Evaluating prediction systems in software project estimation*	**Evaluating prediction systems**	Recently reported scientific evaluations of prediction processes are being re-examined and the initial findings found dangerous.
9.	***S. Malathi et al.***	Quantitative basis for the development and validation	*Analysis of size metrics and effort performance criterion in software cost estimation*	Software Cost Estimation	By constant analysis of different measures and methods, the findings are expected to change.
10.	***A. WINSOR BROWNet al.***	Directed System of Systems (DSOS) or Acknowledged Systems of Systems (ASOS)	*Software cost estimation in the incremental commitment model.*	Software Cost Estimation	The technique is being used to measure the expense of software production of applications

3. CONCLUSION

Software defect has consistently been a functioning & a vastly explorative research zone. Accuracy in software estimation is needed in any software project, not exclusively to appropriately plan the layout, cost, time. Besides this, the budget along with maintenance of the overrun is equally important. In addition to this, sensible approximation needs to be optimally planned as programming associations with enhanced planning as well as estimates shall have the option to get the projects on request. Pre-offered or pre-bid estimation is fundamental in receiving business for an organization. Precision of pre-bid estimation administers the smooth running as well as accomplishment of a task. The production based on judgment frames the reason for resulting project-plan and exercises along with customer responsibilities. As indicated by Porter competitive benefit is accomplished through advancement, quick reaction, and cost-based leadership for clients. Hence, the change in relation to a programming software process is unavoidable and ought to be an ideal state dependent on the measurement of the software programming estimation. Though, the main problem is: Creating historic data important for software, assessment as well as estimation plan need to be inaccessible, accessible yet in accurate or not utilized for software effort & size estimation. The events based on timelines are satisfied for the estimation action, but the resources are not well operational and prepared. Procedures ought to be encouraged to smooth the process of general evaluation strategy

4. 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.

[16] Ochodek, Mirosław, and BatuhanOzgok. "Functional and Non-functional Size Measurement with IFPUG FPA and SNAP—Case Study." In *Software Engineering in Intelligent Systems*, pp. 19-33. Springer, Cham, 2015.

[17] Saroha, Meenakshi, and Shashank Sahu. "Tools & methods for software effort estimation using use case points model—A review." In *International Conference on Computing, Communication & Automation*, pp. 874-879. IEEE, 2015.

[18] Du, Wei Lin, Luiz Fernando Capretz, Ali Bou Nassif, and Danny Ho. "A hybrid intelligent model for software cost estimation." *arXiv preprint arXiv:1512.00306* (2015).

[19] Patil, Lalit V., Sagar K. Badjate, and S. D. Joshi. "Develop Efficient Technique of Cost Estimation Model for Software Applications." International Journal of Computer Applications 87, no. 16 (2014).

[20] Nerkar, L. R., and P. M. Yawalkar. "Software Cost Estimation using Algorithmic Model and Non-Algorithmic Model a Review." *Int J Comput App* 2 (2014): 4-7.

[21] Yiğit, Ferruh, and ÖmerKaanBaykan. "A new feature selection method for text categorization based on information gain and particle swarm optimization." In *2014 IEEE 3rd International Conference on Cloud Computing and Intelligence Systems*, pp. 523-529. IEEE, 2014.

[22] Madheswaran, M., and D. Sivakumar. "Enhancement of prediction accuracy in COCOMO model for software project using neural network." In *Fifth International Conference on Computing, Communications and Networking Technologies (ICCCNT)*, pp. 1-5. IEEE, 2014.