ICIC Express Letters
Volume **15**, Number **10**, October **2021**

ICIC International ©2021 ISSN 1881-803X
pp. **1109–1117**

MODEL AND DATASET TREND IN SOFTWARE PROJECT EFFORT ESTIMATION – A SYSTEMATIC LITERATURE REVIEW

Noptovius Halimawan, Fransiskus Wahyu Panditha Dharma, Wincent and Nico Surantha

Computer Science Department, BINUS Graduate Program – Master of Computer Science
Bina Nusantara University
JL. K. H. Syahdan No. 9, Kemanggisan, Palmerah, Jakarta 11480, Indonesia
{ noptovius.halimawan; fransiskus.dharma; wincent; nico.surantha }@binus.ac.id

Received December 2020; accepted March 2021

Abstract. *As the importance of software development rises in companies, software effort estimation field of study is getting more attention. Accurate estimation is very important to the success of software development projects, but there are no existing models that can serve the purpose perfectly. To contribute in future development of software effort estimation, this systematic literature review analyzes the trends of model and dataset implementation in software effort estimation. The PRISMA methodology is used in the review process. A total of 26 eligible publications are queried and screened from Google Scholar and arXiv for further analysis. The analysis result shows the variability of model implementation in effort estimation. It is also observed in the study that deep learning models are started to be implemented in text-based effort estimation with Long Short-Term Memory (LSTM) and Bidirectional Encoder Representations from Transformers (BERT) model as its base model. Additionally, it is also concluded that dataset utilization in effort estimation does not vary as much as the models.*
Keywords: Software effort estimation, Cost estimation, Software project, Literature review

1. **Introduction.** The advancement of technology in the past years has led to a surge in software development needs [1]. Companies and businesses, including those of non-technological sectors, are facing the transformation of market demands which leads to changes in their technological perspective [2]. The ability of companies to successfully deliver their technological products is a key factor to extend their flexibility in delivering products and services, making it an important catalyst of success.

There are several factors which affect the success of software projects, one of the most important being Software Effort Estimation (SEE). Previous study has found that the capability of accurately estimating a software project's effort leads to a better management of resource and budget [3]. Additionally, it is also found in the study that inability to perform so may lead to loss of jobs and competitive project contracts as most project failures are correlated to its planning phase.

Thorough study has been made on the SEE field of research. The models of SEE are generally divided into algorithmic models, machine learning models, and expert judgement [4]. Mathematical equations and statistical analysis form the basic foundations of algorithmic models. An example is the Constructive Cost Model (COCOMO) [5] which is studied from an extensive size of 63 projects. An improved version of the model named COCOMO-2 [6] has also been published, which is based on a study of 161 projects.

Machine learning models implemented in SEE vary from traditional machine learning to deep learning models. Examples of such models are Support Vector Regression (SVR) models [7], ensemble models [8], and neural networks [4]. Among these models, it has

DOI: 10.24507/icicel.15.10.1109

been found that deep learning models such as neural networks perform generally better than traditional regression models for SEE, showing its effectiveness [1].

Expert judgment method, on the other hand, relies on the decision of knowledgeable experts. This method is argued for being prone to personal interests, therefore vulnerable to subjectivity. Despite so, it has been found that expert judgement is not an entirely bad practice as some projects may benefit from implementing it [9].

Although comprehensive studies have been made in the SEE field of study, there are challenges yet to be solved by researchers. Existing SEE models and methods are still struggling to achieve excellent performance consistently on diverse software projects [4, 10, 11]. Researchers are still studying to improve the achievable accuracy and minimize the loss of SEE models. Additionally, the diverse choice of SEE datasets and evaluation metrics may add the issue in maintaining the consistency of model comparisons.

To support the development of SEE models and contribute in determining the most appropriate SEE model and dataset to implement, we present a thorough Systematic Literature Review (SLR) based on PRISMA methodology. This SLR aims to analyze the trends of the model developments and dataset usage in SEE studies. Publications will be searched from multiple search engines and screened for eligibility based on some predetermined criteria. At the end of the study, we suggest several future challenges that can be studied by researchers in the development of SEE.

2. **Methodology.**

The goal of this SLR is to provide an informative answer to some predetermined research questions. The research questions to be answered in this study are as follows:

1) What are the most trending models to be implemented in SEE cases?
2) What are the most commonly used datasets for training and evaluating SEE models?

To serve as a constructive approach of reviewing, Preferred Reporting Items for Systematic Reviews and Meta-Analyses (PRISMA) [12] methodology is implemented in this study. The study framework is divided into four phases which are identification, screening, eligibility checking, and document inclusion. In the identification phase, several queries will be done on selected article search engines, such as Google Scholar (https://scholar.google.com/) and arXiv (https://arxiv.org/). Then, the query results will be documented and filtered accordingly for suitable studies. The query keywords used in this study are as follows:

1) project effort estimation
2) effort estimation
3) project effort

Upon obtaining the search results, the publications are screened in the screening phase to suit with the study. Several eligibility criteria are used in this screening process, which are:

1) The publications must not be older than five years.
2) The publications must be related to software project effort estimation.
3) The publications must be written in English.
4) The publications must propose and evaluate a model for Software Project Effort Estimation.

In the eligibility checking phase, a collection of 41 feasible publications is produced. These publications are then read thoroughly for content suitability. In the end of the screening process, 26 remaining eligible publications are included (document inclusion phase) in the reviewing pool. The process of this PRISMA-based SLR may be observed in Figure 1.

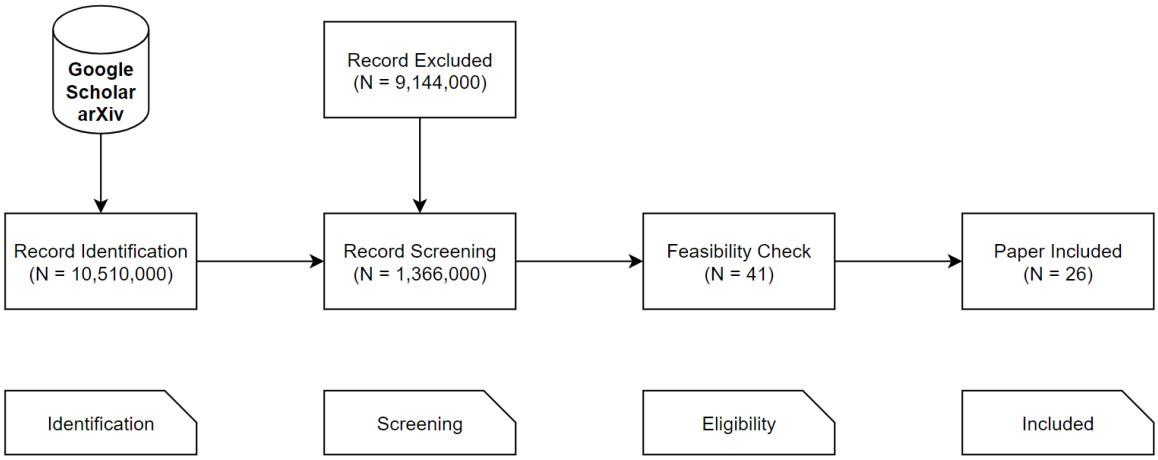

FIGURE 1. PRISMA flowchart

3. Result.

3.1. List of included publications.

Several queries are performed on Google Scholar and arXiv for publication search. The query results show a great number of publications which are published since 2016 with titles or contents related to the search keywords. Due to the impressive number of related publications, only 26 studies are selected and discussed in this review. The results of the document search are as illustrated in Table 1. The detailed list of reviewed studies may also be observed in Table 2.

TABLE 1. Record selection

Source	Found	Screened	Selected
Google Scholar	10,510,000	1,366,000	20
arXiv	152	87	6

3.2. Analysis of publications.

3.2.1. *What are the most trending models to be implemented in SEE cases?* The summary and count of SEE models used in the reviewed studies may be observed in Table 3. As observed in the summary, algorithmic and fuzzy models are frequently utilized in SEE studies. Algorithmic models, as described in the previous section, rely on mathematical equations and theories. On the other hand, the fuzzy model is based on fuzzy logic which is introduced in 1975 [38]. It is a machine learning approach to SEE which excels at handling uncertainty in estimations [30], which is possibly the reason for its high implementation frequency.

Analogy-based models follow with 4 implementation counts. Analogy-based estimation is expert judgement models that relies on equivalent projects. In such models, estimates are derived from similarity measures from these projects. Among these studies, an interesting approach by Resmi and Vijayalakshmi [32] implemented complex machine learning models in the analogy-based model. The authors implemented regressors and classifiers such as Multi-Layer Perceptrons (MLP) to process the dataset, clusters it with expectation maximization, and then applies firefly algorithm for optimization. The result of the model successfully achieved 0.01 MMRE and 97.76 Pred (.25) on the Miyazaki dataset.

Among all SEE model types, the machine learning model most frequently implemented in the studies. These models consist mostly of MLP and optimization models such as the firefly algorithm, humpback whale algorithm, and satin bowerbird optimization.

Table 2. Chosen publications

Source	Year	Title
arXiv	2015	Optimizing Software Effort Estimation Models Using Firefly Algorithm [13]
arXiv	2016	A Hybrid Model for Estimating Software Project Effort from Use Case Points [14]
arXiv	2018	Hyperparameter Optimization for Effort Estimation [15]
arXiv	2019	Software Development Effort Estimation Using Regression Fuzzy Models [16]
arXiv	2020	SE3M: A Model for Software Effort Estimation Using Pre-Trained Embedding Models [17]
arXiv	2020	Software Effort Estimation Using Parameter Tuned Models [18]
Google Scholar	2017	A Light-Weight Incremental Effort Estimation Model for Use Case Driven Projects [19]
Google Scholar	2017	Bayesian Network Model for Task Effort Estimation in Agile Software Development [20]
Google Scholar	2017	Empirical Assessment of Machine Learning Models for Agile Software Development Effort Estimation Using Story Points [21]
Google Scholar	2017	Fuzzy Analogy Based Effort Estimation: An Empirical Comparative Study [22]
Google Scholar	2017	Ontology Based Multiagent Effort Estimation System for Scrum Agile Method [23]
Google Scholar	2017	Satin Bowerbird Optimizer: A New Optimization Algorithm to Optimize ANFIS for Software Development Effort Estimation [24]
Google Scholar	2017	Software Effort Estimation Using Grey Relational Analysis [25]
Google Scholar	2018	An Effective Approach for Software Project Effort and Duration Estimation with Machine Learning Algorithms [26]
Google Scholar	2018	Analogy-Based Model for Software Project Effort Estimation [27]
Google Scholar	2018	Case-Based Reasoning with Optimized Weight Derived by Particle Swarm Optimization for Software Effort Estimation [28]
Google Scholar	2018	Software Effort Estimation Using Grey Relational Analysis with K-Means Clustering [29]
Google Scholar	2018	Uncertainty Management in Software Effort Estimation Using a Consistent Fuzzy Analogy-Based Method [30]
Google Scholar	2019	A Deep Learning Model for Estimating Story Points [31]
Google Scholar	2019	An Effective Software Project Effort Estimation System Using Optimal Firefly Algorithm [11]
Google Scholar	2019	Analogy-Based Approaches to Improve Software Project Effort Estimation Accuracy [32]
Google Scholar	2019	Effort Estimation Model for Software Development Projects Based on Use Case Reuse [33]
Google Scholar	2019	Process-Driven Incremental Effort Estimation [34]
Google Scholar	2020	Exploring the Whale Optimization Algorithm to Enhance Software Project Effort Estimation [35]
Google Scholar	2020	Hyperparameters Tuning of Ensemble Model for Software Effort Estimation [36]
Google Scholar	2020	Validation of Existing Software Effort Estimation Techniques in Context with Mobile Software Applications [37]

Table 3. Utilized models in publications

Model	Count	Publications
Algorithmic	5	[19, 25, 29, 33, 34]
Fuzzy	5	[11, 16, 22, 30, 32]
Multi-layer perceptron	5	[16, 26, 32, 36, 37]
Analogy-based	4	[22, 27, 30, 32]
Firefly algorithm	3	[11, 13, 32]
Grey relational analysis	3	[25, 28, 29]
Bayesian	2	[20, 34]
Deep learning	2	[17, 31]
Ensemble	2	[26, 36]
Genetic algorithm	2	[36, 37]
Particle swarm optimization	2	[28, 36]
Random forest	2	[21, 36]
Support vector machine	2	[14, 26]
BERT	1	[17]
CART	1	[15]
Humpback whale algorithm	1	[35]
Ontology-based	1	[23]
Partial least square	1	[18]
Satin bowerbird optimization	1	[24]

Traditional machine learning models such as the Support Vector Machine (SVM) have been implemented in SEE studies in the early years of the reviewed studies [14]. It is also observable in the studies that deep learning models such as MLP are implemented on later studies in 2018 [26].

It is only until 2019 that more sophisticated deep learning models such as the Long Short-Term Memory (LSTM) model are implemented in the observed studies [31]. In the same study, LSTM is used to learn from issue report text to estimate story points in agile projects which changes the perspective of the study to a Natural Language Processing (NLP) case. The deep learning model implementation in 2020 by Fávero et al. [17] progresses to the use of Bidirectional Encoder Representations from Transformers (BERT) [39] which is the state-of-the-art text classification model. BERT is one of the first breakthrough models to successfully implement transfer learning in NLP. The study implements SEE from software requirement documents of open source projects with BERT, achieving 4.25 Mean Absolute Error (MAE). Though deep learning is a rapidly growing field of study, not much has been implemented in SEE.

Aside from deep learning, the role of biologically-inspired optimization algorithms in machine learning SEE models is also noticeable from the studies. These algorithms include the implementation of firefly algorithm, humpback whale algorithm, and satin bowerbird optimization. The firefly algorithm [11] is inspired by the characteristics of a firefly being attracted to light. The closer the firefly to the objective, the brighter it will shine and the higher the chance of it to attract nearby fireflies. The satin bowerbird optimization [24], meanwhile, is inspired by the courtship or mating phase of the bird. The male satin bowerbird has the characteristic to build bowers for attracting its mating partner and mimic its competitors. Finally, the humpback whale algorithm [35] is inspired by the spiral pattern the whales make upon attacking their prey.

Based on the model utilization, it can also be concluded that model variability is high in the study of SEE. Judging from the sorted model frequencies, the median value of

implementation frequency is a mere 2 from 26 studies. Therefore, it is more likely for future studies to explore newer approaches compared to improving the current method.

3.2.2. *What are the most commonly used datasets for training and evaluating SEE models?* The summary of dataset utilization in the studies is observable in Table 4. As illustrated in the table, most studies use the ISBSG dataset or choose to self-provide the dataset from education or industry projects. The ISBSG dataset is a repository of diversified instances which require preprocessing before usage in SEE [36].

Table 4. Utilized datasets in publications

Dataset	Count	Publications
ISBSG	7	[15, 16, 22, 24, 26, 30, 36]
Self-provided	7	[14, 19, 20, 21, 23, 33, 34]
Albrecht	6	[15, 18, 22, 24, 30, 32]
COCOMO	6	[11, 18, 22, 29, 30, 32]
Desharnais	6	[15, 18, 22, 28, 30, 32]
NASA	6	[11, 13, 18, 30, 32, 35]
Kemerer	5	[15, 22, 24, 25, 32]
Maxwell	5	[15, 18, 27, 28, 32]
China	4	[15, 18, 22, 30]
Miyazaki	4	[15, 22, 30, 32]
Kitchenham	2	[15, 18]
Open source with JIRA	2	[17, 31]
Finnish	1	[15]
SAMOA dataset	1	[37]
UCP dataset	1	[18]

Interestingly, it can be observed that the ISBSG, Albrecht, and Desharnais datasets are implemented on machine learning and fuzzy models in the reviewed studies. For example, the study by Moosavi and Bardsiri [24] enhanced the Adaptive Neuro-Fuzzy Inference System (ANFIS) model with satin bowerbird optimization on the Albrecht, Kemerer, and ISBSG datasets to achieve 0.49 MMRE and 0.637 Pred (0.25). However, the same does not apply to algorithmic models in the reviewed publications. It has been observed that three out of five reviewed studies that use algorithmic models provide their own SEE dataset [19, 33, 34]. For example, Qi and Boehm [34] used 61 student projects to form their dataset and relied on Jira tickets and reports to determine the project effort. To build their models, the entire observed publications which have proposed deep learning models use open source projects, also with Jira reports, for their datasets [17, 31].

Most of the datasets in the SEE study are found in repositories such as the SEACRAFT and the PROMISE repository. The SEACRAFT repository which is used by Xia et al. [15] contains most of the datasets in the reviewed SEE studies, such as the Kemerer, Albrecht, ISBSG, Finnish, Miyazaki, Maxwell, Desharnais, Kitchenham, and China datasets. It can be concluded from this observation that SEE dataset choice pattern exists, but it does not vary as much as its model variations and progressions.

4. **Future Challenge.** A large variety of models have been developed and tested in the SEE field of study. Though some has proven to show state-of-the-art performance, there are still improvements to be made and fields to be explored. Among all of the evaluated models in SEE, not much attention has been given to deep learning models. Deep learning models are proven to be better than traditional machine learning models on various tasks

such as NLP and computer vision [40]. Its superior performance promises a great potential of improvement in SEE.

Aside from deep learning, more sophisticated NLP study can also be done in SEE. Some of the observed studies in this review have developed their models using text datasets from open source projects with Jira [17, 31]. This means that aside from developing the model solely for optimization of variables, the feature extraction process from the analyzed project text can also be focused on. Implementing more state-of-the-art NLP models may also be an interesting challenge in future studies. For example, the Robustly optimized BERT approach (RoBERTa) model is a promising choice. The model has been found to exceed the performance of BERT and other state-of-the-art baseline NLP models [41]. As it has only recently been studied, future SEE studies may benefit from such improvements on text-based effort estimation.

5. **Conclusion.** This systematic literature review implements the PRISMA methodology to analyze the trends of model and dataset implementation in software effort estimation. Related publications are queried through Google Scholar and arXiv search engines, which will then be screened for eligibility based on predetermined criteria. After the screening, a total of 26 publications are reviewed for this study. The result of model implementation analysis shows some variability of model choice. Compared to the expert judgement and algorithmic models, machine learning models are the most frequently implemented in SEE. Deep learning models may also be found in the later years of the study, which is also the start of text-based SEE studies. Additionally it is also observed that the dataset utilization of SEE does not vary as much as the models. Much can still be studied in software effort estimation, especially in the deep learning model development.

Acknowledgment. This publication is fully supported by Bina Nusantara University.

REFERENCES

[1] I. F. de Barcelos Tronto, J. D. S. da Silva and N. Sant'Anna, Comparison of artificial neural network and regression models in software effort estimation, *2007 International Joint Conference on Neural Networks*, pp.771-776, 2007.

[2] M. Rachinger, R. Rauter, C. Müller, W. Vorraber and E. Schirgi, Digitalization and its influence on business model innovation, *Journal of Manufacturing Technology Management*, vol.30, pp.1143-1160, 2019.

[3] J. G. Borade and V. R. Khalkar, *Software Project Effort and Cost Estimation Techniques*, Tech. Rep., 2013.

[4] P. S. Kumar, H. Behera, A. Kumari K, J. Nayak and B. Naik, Advancement from neural networks to deep learning in software effort estimation: Perspective of two decades, *Computer Science Review*, vol.38, DOI: 10.1016/j.cosrev.2020.100288, 2020.

[5] B. W. Boehm, Software engineering economics, *IEEE Trans. Software Engineering*, vol.SE-10, pp.4-21, 1984.

[6] P. Musilek, W. Pedrycz, N. Sun and G. Succi, On the sensitivity of COCOMO II software cost estimation model, *Proc. of the 8th IEEE Symposium on Software Metrics*, pp.13-20, DOI: 10.1109/METRIC.2002.1011321, 2002.

[7] P. L. Braga, A. L. I. Oliveira and S. R. L. Meira, A GA-based feature selection and parameters optimization for support vector regression applied to software effort estimation, *Proc. of the 2008 ACM Symposium on Applied Computing (SAC'08)*, 2008.

[8] E. Kocaguneli, T. Menzies and J. W. Keung, On the value of ensemble effort estimation, *IEEE Trans. Software Engineering*, vol.38, pp.1403-1416, 2012.

[9] R. T. Hughes, Expert judgement as an estimating method, *Information and Software Technology*, vol.38, pp.67-75, 1996.

[10] P. Rijwani and S. Jain, Enhanced software effort estimation using multi layered feed forward artificial neural network technique, *Procedia Computer Science*, vol.89, pp.307-312, 2016.

[11] V. Resmi, S. Vijayalakshmi and R. S. Chandrabose, An effective software project effort estimation system using optimal firefly algorithm, *Cluster Computing*, vol.22, pp.11329-11338, 2019.

[12] D. Moher, A. Liberati, J. Tetzlaff and D. G. Altman, Preferred reporting items for systematic reviews and meta-analyses: The PRISMA statement, *PLoS Medicine*, vol.6, 2009.
[13] N. Ghatasheh, H. Faris, I. Aljarah and R. M. H. Al-Sayyed, Optimizing software effort estimation models using firefly algorithm, *Journal of Software Engineering and Applications*, vol.8, pp.133-142, 2015.
[14] M. Azzeh and A. B. Nassif, A hybrid model for estimating software project effort from use case points, *Applied Soft Computing*, vol.49, pp.981-989, 2016.
[15] T. Xia, R. Krishna, J. Chen, G. Mathew, X. Shen and T. Menzies, Hyperparameter optimization for effort estimation, *arXiv.org*, arXiv: 1805.00336, 2018.
[16] A. B. Nassif, M. Azzeh, A. Idri and A. Abran, Software development effort estimation using regression fuzzy models, *Computational Intelligence and Neuroscience*, 2019.
[17] E. M. D. B. Fávero, D. Casanova and A. R. Pimentel, SE3M: A model for software effort estimation using pre-trained embedding models, *arXiv.org*, arXiv: 2006.16831, 2020.
[18] A. Baghel, M. Rathod and P. Singh, Software effort estimation using parameter tuned models, *arXiv.org*, arXiv: 2009.01660v1, 2020.
[19] K. Qi and B. W. Boehm, A light-weight incremental effort estimation model for use case driven projects, *2017 IEEE the 28th Annual Software Technology Conference (STC)*, pp.1-8, 2017.
[20] S. Dragicevic, S. Celar and M. Turic, Bayesian network model for task effort estimation in agile software development, *Journal of Systems and Software*, vol.127, pp.109-119, 2017.
[21] S. M. Satapathy and S. K. Rath, Empirical assessment of machine learning models for agile software development effort estimation using story points, *Innovations in Systems and Software Engineering*, vol.13, pp.191-200, 2017.
[22] A. Idri and I. Abnane, Fuzzy analogy based effort estimation: An empirical comparative study, *2017 IEEE International Conference on Computer and Information Technology (CIT)*, pp.114-121, 2017.
[23] M. Adnan and M. Afzal, Ontology based multiagent effort estimation system for scrum agile method, *IEEE Access*, vol.5, pp.25993-26005, 2017.
[24] S. H. S. Moosavi and V. K. Bardsiri, Satin bowerbird optimizer: A new optimization algorithm to optimize ANFIS for software development effort estimation, *Engineering Applications of Artificial Intelligence*, vol.60, 2017.
[25] M. Padmaja and D. Haritha, Software effort estimation using grey relational analysis, *International Journal of Information Technology and Computer Science*, vol.9, pp.52-60, 2017.
[26] P. Pospieszny, B. Czarnacka-Chrobot and A. Kobylinski, An effective approach for software project effort and duration estimation with machine learning algorithms, *Journal of Systems and Software*, vol.137, pp.184-196, 2018.
[27] Ardiansyah, M. M. Mardhia and S. Handayaningsih, Analogy-based model for software project effort estimation, *International Journal of Advances in Intelligent Informatics*, vol.4, pp.251-260, 2018.
[28] D. Wu, J. Li and C. Bao, Case-based reasoning with optimized weight derived by particle swarm optimization for software effort estimation, *Soft Computing*, vol.22, pp.5299-5310, 2018.
[29] M. Padmaja and D. Haritha, Software effort estimation using grey relational analysis with k-means clustering, in *Information Systems Design and Intelligent Applications. Advances in Intelligent Systems and Computing*, V. Bhateja, B. Nguyen, N. Nguyen, S. Satapathy and DN. Le (eds.), Singapore, Springer, 2018.
[30] S. Ezghari and A. Zahi, Uncertainty management in software effort estimation using a consistent fuzzy analogy-based method, *Applied Soft Computing*, vol.67, pp.540-557, 2018.
[31] M. Choetkiertikul, H. K. Dam, T. Tran, T. Pham, A. Ghose and T. Menzies, A deep learning model for estimating story points, *IEEE Trans. Software Engineering*, vol.45, no.7, pp.637-656, 2019.
[32] V. Resmi and S. Vijayalakshmi, Analogy-based approaches to improve software project effort estimation accuracy, *Journal of Intelligent Systems*, vol.29, pp.1468-1479, 2019.
[33] K. Rak, Ž. Car and I. Lovrek, Effort estimation model for software development projects based on use case reuse, *Journal of Software: Evolution and Process*, vol.31, 2019.
[34] K. Qi and B. W. Boehm, Process-driven incremental effort estimation, *Proceedings – 2019 IEEE/ACM International Conference on Software and System Processes (ICSSP2019)*, pp.165-174, 2019.
[35] A. A. Fadhil and R. G. Alsarraj, Exploring the whale optimization algorithm to enhance software project effort estimation, *2020 the 6th International Engineering Conference "Sustainable Technology and Development" (IEC)*, pp.146-151, 2020.
[36] S. K. Palaniswamy and R. Venkatesan, Hyperparameters tuning of ensemble model for software effort estimation, *Journal of Ambient Intelligence and Humanized Computing*, 2020.

[37] M. Pandey, R. Litoriya and P. Pandey, Validation of existing software effort estimation techniques in context with mobile software applications, *Wireless Personal Communications*, vol.110, pp.1659-1677, 2020.
[38] L. A. Zadeh, Fuzzy logic and approximate reasoning, *Synthese*, vol.30, nos.3-4, pp.407-428, 1975.
[39] J. Devlin, M.-W. Chang, K. Lee and K. Toutanova, BERT: Pre-training of deep bidirectional transformers for language understanding, *Proc. of NAACL-HLT 2019*, pp.4171-4186, 2019.
[40] Y. LeCun, Y. Bengio and G. Hinton, Deep learning, *Nature*, vol.521, pp.436-444, 2015.
[41] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer and V. Stoyanov, RoBERTa: A robustly optimized BERT pretraining approach, *arXiv.org*, arXiv: 1907.11692v1, 2019.