An Overview of Machine Learning Approaches to Software Development Cost Estimation

Mohammed Maher
dept. of Software Engineering
University of Mosul
Mosul, Iraq
maher@uomosul.edu.iq

Jamal Salahaldeen Alneamy
dept. of Software Engineering
University of Mosul
Mosul, Iraq
jamal_alneamy@uomosul.edu.iq

***Abstract*— Software cost estimation still presents a real challenge to software development firms and practitioners. One of the crucial activities in project management is software cost estimation. The accuracy of the estimated cost has a critical impact on the project's success or failure. Overestimation and underestimation are serious challenges to project managers, especially in today's competitive marketplace. Inaccurate estimation may lead to over budget, missing deadlines, business loss, and even project failure. The precise cost estimation will enable the project manager to forecast and allocate the necessary resources to deliver a successful product. Many researchers have published research papers in this area. In this paper, the study's goal is to outline the recent machine learning approaches to software cost estimation. The study will address the problems with conventional cost estimation models and provide a comprehensive review of the researches regarding software cost estimation with machine learning. The current study will discuss the intelligence models for software cost estimation and highlight the models that excel in this area.**

Keywords*— *Software cost estimation, Machine learning, Software cost estimation model, Artificial Intelligence, COCOMO

I. INTRODUCTION

Software cost estimation is an open research problem. An enormous number of researchers have published papers in this domain, each one claims achieving an accurate estimation. An essential step in the software engineering process is software cost estimation. An accurate estimation raises the likelihood that a project will succeed and lowers the risk of failure. Also, it helps the project manager to determine the necessary resources to deliver the project on time within budget and according to the predefined specification. Projects with inaccurate estimation may face problems meeting their deadlines, being over budget, not delivering qualified products, customer dissatisfaction, business loss, and project abortion or failure [1]. There is no control over the process of developing software, and the real cost and effort of the project are higher than what was anticipated, according to research done on software cost and effort estimation. As a result, most projects run over budget and miss deadlines. The Standish Group International Chaos Report of 2015 stated that 56% of projects were over budget, and 60% of the projects had missed their scheduled deadlines. A similar study by the Project Management Institute in 2017 reported that 43% did not finish within the predefined estimated budget, 48% of the projects were late, and 32% had been aborted or failed because of the budget loss [2].

Software development companies and teams use software engineering cost models and estimation techniques for budgeting which represents the primary use, trade-off and project control and planning, analysis of the risk, and software improvement investment analysis. Since the 1960s, much research has been started with an extensive study of software development cost estimation, which led to the development of some useful models such as: SLIM, Checkpoint, PRICES, SEER, and COCOMO. But as the software's scope and relevance increased, so its complexity increases, making it extremely challenging to effectively determine the cost of creating software, each of these models encountered the same problem [3]. Various methods and models for software cost assessment have been developed during the past few years. Non-algorithmic and algorithmic models are the two basic groups into which these models can be divided [4]. The algorithmic models use statistics and mathematical equations to calculate the costs. These models make advantage of factors including: project size, software kind, and software development methodology. Software engineers and project managers have employed algorithmic models, also referred to as conventional models, in a variety of projects. However, by conducting some research in this area, it can be seen that each technique will estimate the cost and work of software development differently. This provides a challenge for project managers, especially when it comes to managing the project's resources [5].

The mathematical techniques for cost estimate are the foundation of the algorithmic cost model. This model uses a variety of estimated approaches to determine the cost depending on the method being used, project's size, software team, kind of software, and software attributes. The COCOMO is a very popular algorithmic cost model for software. Boehm brought the COCOMO 81 Model in 1981. The COCOMO-II model, which is currently in use, bases the effort estimation in software projects on Person-Month (PM). This model consists of five scale factors and seventeen multipliers, and it measures size in terms of function points or lines of code [6].

Traditional cost estimation methods require work to document activities, which complicates and lengthens the estimation process. Indeed, these models call for a number of initial inputs, which is challenging, particularly early in the process of developing software. Additionally, a number of additional characteristics, including the interaction between them, and the experience of the software engineers, as well as the team's development in the relevant business area, are not always simply and exactly predicated [7].

Due to the shortcomings of traditional models, the idea of models based on machine learning emerges. The capacity of machine learning powered techniques for software cost assessment to learn and change their behavior on their own makes them good alternatives. Multi-layer Perceptron, Support Vector Machine, Extreme Machine Learning, Random Forest, Decision Tree, Artificial Neural Network, Bayesian Network, K-Nearest Neighbors, and Recurrent

XXX-X-XXXX-XXXX-X/XX/$XX.00 ©20XX IEEE

Neural Network are some of the methods that have been employed in machine learning to forecast the price of software development [8]. In order to detect the existed association between the main cost factors at the entry side and the cost of development effort at the other side, machine learning-powered approaches can be applied [9].

In this paper, the main objective is to outline the recent methods for software cost prediction based on ML techniques and to highlight the models that provide accurate cost estimations and excel in comparison to conventional models. The remainder of the paper is organized as follows: Section 2 surveyed the recent research in the field of cost and effort prediction for software development. Section 3 overviews the main software engineering repositories used to build the machine learning models. Section 4 lists briefly the most used soft computing techniques for estimating cost. Discussion of the results and the conclusions are in sections 5 and 6 respectively.

II. RELATED WORKS

The software project is considered as one of the riskiest projects in the field of contemporary industry. Changing customer requirements, software development tools, and methodologies, and the intangible nature of the software can seriously affect the value of the project schedule, the budget and the estimated cost, and the quality of the delivered product. It can be noticed that the risk of successful completion of a software product could have consequences of a specific hazard [10]. Enormous research regarding software cost estimation using machine learning has been performed. Thus, this section discusses the recent studies and highlighting the models and techniques they have followed.

Nevena et al. [11] presented two distinct artificial neural networks (ANN) topologies as a useful tool for anticipating and calculating software development efforts. By using Orthogonal Array (OA), they sought to decrease the error in the predictions of effort. They also sought to identify the most straightforward artificial neural network architecture for improved learning. Authors in [12] have put into practice a multilayer perceptron with a back propagation method for estimating software work. For the network's training, they used the Maxwell dataset. The results they had gotten using the linear regression technique were then compared.

A variety of strategies for preparing data, including transformation, cleaning and selection, have been used by Suyash et al. [13] with machine learning approaches such as: Recurrent Neural Network (RNN), Probabilistic and Multi-Layer Neural Network to improve the quality of the data set provided to the model. To enhance the precision of the software cost and effort prediction process, they created an ensemble method.

The authors of [14] have combined distributed software project datasets with genetic algorithms, neural networks, and genetic programming. They created a number of cost estimation models, which the COCOMO 81 dataset was used to validate. The suggested models can give software project managers abilities that they can use to obtain more accurate software cost estimations. Ming Qin [15] in his paper, created a method for machine learning that uses the Conditional Random Field and Lattice Long Short-Term Memory. The framework was put up as a solution to the issues with FP-based (Function Points based) software cost estimation. The named entity recognition and categorization challenges are created from the framework's transformation of the FP identification task. Additionally, lattice LSTM (Long Short-Term Memory), which benefits from both Chinese symbols and word knowledge, is first used in the field of software cost calculation. Lattice LSTM completes the basic NER work by extracting the named item that is connected with the FP. CRF is used to carry out the work of FP type categorization in order to further increase the precision of FP recognition.

Kui Zhang et al. [16] have developed a named entity recognition (NER) model based on deep learning as opposed to manual function point recognition. To specify new requirements in the same area, a bidirectional long short-term memory (LSTM) and a conditional random field (CRF) model were trained on requirements which have been labeled from the software sector. Twenty-nine genuine projects were used to validate the suggested paradigm. For the quantitative assessment of the suggested estimating function points with NER models's efficiency enhancement, a comparative experiment was created. According to the outcome, the validity of the function point analysis using bidirectional (LSTM) and a conditional random field on samples of the test significantly improved for the NER model. Shaina and Nidhi [17] suggested a model that computes the software cost estimation using a single layer ANN. Sixteen input parameters are used in this model, including fifteen effort multipliers and one bias weight. Before applying the identical activation function, they determined the net input output (sum of each input neuron's product to its associated weight). Authors in [18] suggested a Flower Pollination Algorithm (FPA) to use a typical Turkish industry dataset to make a Constructive Cost Model II's parameters as optimal as possible (COCOMO-II). As shown by mean magnitude of relative errors (MMRE) and Manhattan distance, experimental results show that the suggested method provides an improved prediction when contrasted to current methods like the classic COCOMO-II and bat algorithm. Mukesh et al. [19] employed genetic programming to estimate software work. Individuals must evolve during implementation in order to produce the optimal results over numerous generations. Based on the review of the literature, the metrics are selected to assess the model. This project makes use of common software engineering datasets in order to realize suitability and any emerging relationships. To generate more trustworthy evaluation summaries, they used K Fold validation.

Mahmood et al. [1] have examined and evaluated 13 methods based on machine learning using five major statistical indices. IBK, Random Forest, SMOreg RepTree, Multilayer Perceptron, Zero-R, Decision Table, Linear Regression, M5P, Additive Regression, Gaussian Processes, K-Star, and Input Mapped Classifier are some of the techniques or classifiers that are employed. They have made use of publicly accessible datasets that have actual data from software engineering. In their trial, Random Forest produced the best outcomes. Multi-Layer Perceptron Neural Network Model and other ensemble models were investigated by the author in [21] to enhance the functionality of the software effort estimation method. In order to discover the model that best fits the Desharnais dataset, they constructed the Bagging-MLPNN, Lasso-MLPNN, Ridge-MLPNN, MLPNN, and AdaBoost-MLPNN models. Then, based on R squared, the results of different models were compared.

An estimation model for predicting story points was set-up by Morakot et al. [22] and was built on a creative fusion of

two potent deep learning techniques: memory recurrent highway network and long short-term. Without any manual feature engineering, their estimation model could be trained from the beginning to the end, from the raw input data to the predicted outputs. Empirical analysis reveals that their approach regularly performs better on the basis of Standardized Accuracy and Mean Absolute Error than three baselines for common effort estimation and two alternatives. In a study that is similar to the current one, Hung and Ali [23] suggested a method based on Graph Neural Networks, a novel strategy that has been applied to text classification in natural language processing. Graph Neural Networks' advantages stem from their ability to learn information using graph data structures, which have more representatives than methods of vectorizing word sequences, such as the relationships between words. In their study, they demonstrated the potential and potential difficulties of text classification using Graph Neural Networks for estimating story points. According to the experiments, the GNN Text Level Classification is comparable to the conventional method in that it can classify story points with an accuracy of roughly 80%.

III. SOFTWARE ENGINEERING DATASETS

A collection of precise, practical, and relevant real-world data regarding software projects is referred to as a software engineering data repository. These datasets contain descriptive and quantitative data regarding resources, goods, services, procedures, managing, etc. Known enterprises, as well as particular software companies and researchers, are gathering such data for a variety of reasons. These datasets are helpful for undertaking benchmark, exploratory, and comparative literature in the majority of scientific and technical fields [24]. The ISBSG (International Software Benchmarking Standards Group) Repository and PROMISE (PRedictOr Models In Software Engineering) are the two primary repositories for software projects in the world of software engineers.

A. ISBSG

The ISBSG's data repository is an openly accessible dataset with data on software projects that was gathered from numerous enterprises throughout the globe starting in 1989 [25]. Several studies concentrating on software estimation have used this data set. The ISBSG offers information about:

- Defect prediction: factors include the quantity of flaws reported over the diverse phases of the life cycle of software development (SDLC), the effort involved, the amount of the code in scale of function points and lines, the quantity of specifications rework made throughout the life cycle of software development (SDLC), what kind of program it is, etc.
- Prediction of effort includes stages of effort, a work effort summary, a consistent level of effort.

B. PROMISE

Sayyad and Menzies have started the PROMISE Software Engineering Repository in December 2004 to promote the creation of estimation frameworks for software engineering projects. This repository's initial iteration was produced using a dataset of NASA and kept at the School of Information Technology and Engineering, University of Ottawa, Canada. These datasets are gathered in this repository, that the software engineering community has made freely available to the public in order to assist researchers and the software industry. Depending on the topic covered, PROMISE members have divided the datasets into 5 groups. In fact, there are 12 datasets in the effort and cost prediction category [26].

IV. SOFTWARE COST ESTIMATION MODELS

Soft computing models like Neural Network, Decision Tree, Genetic Algorithm, Machine Learning and Hybrid Models are being widely applied for software development cost and effort estimation due to their good generalization capability, fast learning speed and they are easier to be designed. Furthermore, they assist in decision-making using minimal human interference based on data analysis. Below are the top commonly used machine learning models:

A. Artificial Neural Networks

The organic human brain is mimicked by an artificial neural network (ANN). Various academics have presented neural network powered algorithms for estimating software development effort. The importance of the neural network powered approach for estimating software effort is emphasized in the majority of articles. Since the last two decades, neural network-based models have receiving a lot of attention in the software effort estimation task [27].

B. Support Vector Regression

A machine learning technique that has proved effective for estimating the cost and effort of software development. SVR's flexibility to be adjusted to various and diversified bits of data is what makes it so powerful. Regression using support vectors and high-dimensional feature spaces benefit from the support vector machine model. This method looks for a boundary that divides the data according to the desired feature. Thus, a method based on SVMs called support vector regression is appropriate for prediction issues [28].

C. Decision Tree

One of the most used methods for creating prediction models is the decision tree. Classification Tree and Regression Tree are the two categories into which it can be classified. When the output class is an observation from an ordinal or nominal data class, the classification tree is employed. When the output value to be predicted is from the interval domain, Regression Trees are employed [29].

D. Random Forest

A supervised learning technique called "Random Forest" builds a forest from an ensemble of decision trees and introduces some element of randomness into it. To produce more precise and reliable predictions, Random Forest builds many decision trees and blends them [20].

E. Case-Based Reasoning

The approach of assessing the cost based on machine learning that has received the most research is called case-based reasoning (CBR), which uses an analogy approach. CBR is the most often applied methodology in software production because of its intellectual straightforwardness and quantifiable dependability. In 1997, a CBR implementation for software effort estimation was initially proposed. Numerous research results point to the CBR methodology as the best method for determining the expense and effort involved in software development [2].

TABLE I. COMPARISON OF MACHINE LEARNING-BASED TECHNIQUES FOR SOFTWARE COST ESTIMATION

Paper	Model	Dataset	Performance Error	Year
Carvalho et al. [8]	Extreme Learning Machine	Desharnais	MMRE = 0.18	2021
Rankovic et al. [11]	Artificial Neural Networks	COCOMO 2000, COCOMO 81, NASA and Kemerer	MMRE = 0.52	2021
Ullah et al. [18]	Flower Pollination	Standard Turkish industry dataset	MMRE = 34.2	2019
Mahadev et al. [19]	Genetic Programming	China	MMRE = 11.9	2020
Arias et al. [28]	Support Vector Machine	ISBSG	MMRE = 0.84	2020
Abdelalia et al. [30]	Random Forest	ISBSG, Tukutuku and COCOMO	MMRE = 1.29	2020
Dizaj et al. [31]	Genetic with Bat Algorithm	NASA and Kemerer	MMRE = 17.9	2018
Christina et al. [32]	Neuro Fuzzylogic	NASA	MMRE = 26.9	2019
Ahmad et al. [33]	Whale Crow Optimization	COCOMO 2000, COCOMO 81, NASA and Desharnais	MMRE = 0.24	2019
Fadhil et al. [34]	Dolphin Algorithm	NASA	MMRE = 14.5	2020

F. Extreme Learning Machine

The ELM has been employed in the latest research papers to predict effort and expense. Guang Bin Huang came up with the ELM algorithm. With a few minor variations, its building is analogous to a feed forward neural network with one layer. The output layer's neurons' weights are calculated analytically, but the initial weights of the input layer's neurons are created at arbitrarily rather than being changed [7].

V. RESULTS AND DISCUSSION

The findings from examining the research papers included in this article are covered in this part. On two datasets, Al Asheeri et al. [1] used 13 ML methods. They employed the following evaluation standards in their work: RRSE MAE, RMAE, RAE, and R2. The suggested approach seeks to forecast the effort using dataset features and contrast them with the actual measures used to determine the inaccuracy using various metrics. In the first trial, Random Forest and three other models—Additive Regression, REP-Tree, and K-star got the greatest results. The ZeroR approach had the worst performance on the initial dataset, whereas the Multilayer Perceptron, IBk, and Linear Regression methods had poor outcomes on the second dataset.

To calculate the software effort, Carvalho et al. [8] used the Desharnais datasets and five machine learning algorithms such as: Extreme Learning Machine (ELM), Linear Regression (LR), Support Vector Machine (SVM), Multilayer Perceptron (MLP), K-and Nearest Neighbors (kNNs).Using the ELM model, they were able to produce the best result with the fewest errors.

In order to discover the best solution, Rankovic et al. [11] employed two Artificial Neural Networks (ANN) architectures using Orthogonal Array Tuning Method (OATM), which produces successful results with considerably reduced multiple experiments. On three benchmark datasets, the proposed OATM is tested on three separate tasks. The current study's findings demonstrate that, when applied in three stages of the assessment—testing, validation, and training—the pair Neural Network designs indicated above perform better in obtaining the least MMRE when evaluating software effort than when using conventional techniques.

Four machine learning methods were utilized by Shukla et al. [13]: LR, SVM, KNN, and MLP. 67 percent of the examples from the Desharnais dataset were utilized for the training of these models, while the remaining 33 percent were used to test their effectiveness. The MLPNN technique outperformed the other three models employed in their study, with 79 percent of successful estimations and there was a difference of 6-7 percent with other models, they found after examining the performance of LR, SVM, KNN, and MLPNN.

A stranded dataset from a software project for the Turkish industry was used by Ullah et al. [18] to optimize the COCOMO-II technique's current coefficients. Their suggested work's major objective is to use the FPA's results for better cost estimation to optimize the COCOMO-II model's present coefficients: A and B. Based on the study of the results, they concluded that applying the FPA optimization technique to COCOMO-II produced results for estimated costs that were greater than those for COCOMO-II as a whole.

Software effort estimation has been done using genetic programming by Mahadev et al. (19). Individuals must evolve during implementation in order to produce the optimal results over numerous generations. For the purpose of evaluating the prediction model, the genetic programming model implementation utilizing Pred25 and MMRE metrics was taken into consideration. In order to evaluate the predictive model, various datasets of differing sizes were used. The outcomes reveal that the GP (Genetic Programming) model exhibits great precision values. How big is the dataset is another element that affects how accurately the results deviate across many folds of validation. When a lesser dataset is used in comparison to larger ones, a considerable degree of variance is seen. The findings support the additional investigation of GP's various qualities for improved outcomes.

Support vector regression (SVR) was used in conjunction with this random search hyper-parameter tuning strategy by Arias et al. [28] to estimate software effort. Results were contrasted with those from grid search-tuned models and ridge regression models. The performance of the hyper-parameter optimized SVR model was noticeably superior to other regression techniques.

VI. Conclusions

Software cost estimation and the early stage of the software development process is an activity that is crucial for the successful delivery of the software product. An accurate estimation enables the efficient management of the development process. It allows the project manager to plan, budget, forecast, and accurately allocate the necessary resources for the project. Unlike other industries, cost estimation is not an easy process because of the nature of the software. The current study surveyed several research papers in the domain of cost estimation techniques based on machine learning. Many intelligent techniques have been used in the studied papers, but Artificial Neural Networks, Support Vector Regression (SVR), Decision Tree, Random Forest, Case-Based Reasoning and Extreme Learning Machine (ELM) are the broadly used models. Most of the papers obtained the datasets for the models from two software engineering repositories, ISBSG, and PROMISE. In this study, it has been concluded that soft computing models have fine generalization capability, and they give fairly accurate estimates in comparison with conventional models. After analyzing results produced by the presented models, Extreme Learning Machine (ELM) outperformed the other techniques. This paper is part of ongoing research in software cost estimation. In the future, the presented models can be evaluated against a unified datasets, and compare the results with other models.

References

[1] Mahmood M. Al Asheeri, and Mustafa Hammad, “Machine Learning Models for Software Cost Estimation” in 2019 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies, 2019.

[2] Yasir Mahmood, Nazri Kama, Mazlan Ali, and Azri Azmi, “Improving Estimation Accuracy Prediction of Software Development Effort: A Proposed Ensemble Model” in the 2nd International Conference on Electrical, Communication and Computer Engineering (ICECCE), Istanbul, Turkey, 2020.

[3] Barry Boehma, Chris Abts and Sunita Chulani, "Software development cost estimation approaches – A survey" in Annals of Software Engineering 10, 2000, pp. 177–205.

[4] Rijwani, Poonam, Sonal Amit Jain, and Sonal Jain. "Software Effort Estimation Development From Neural Networks to Deep Learning Approaches." Journal of Cases on Information Technology (JCIT) 24, pp. 1-16, 2022.

[5] Farhad Akhbardeh and Hassan Reza, "A Survey of Machine Learning Approach to Software Cost Estimation" in 2021 IEEE International Conference on Electro Information Technology (EIT), pp. 405–408, 2021.

[6] Zaineb Sakhrawi, Asma Sellami and Nadia Bouassida "Software Enhancement Effort Estimation using Machine Learning Regression Methods" in International Journal of Computer Information Systems and Industrial Management Applications, Volume 12 pp. 412-423, 2020.

[7] Mayank Jha and Richa Jha, "Comparing the Effort Estimated By Different Models" in 2020 6th International Conference on Advanced Computing & Communication Systems (ICACCS), pp. 1148-1154, 2020

[8] Halcyon Davys Pereira De Caevalho, Fagundes R.,and S. Wylliams, "Extreme Learning Machine Applied to Software Development Effort Estimation" in IEEE Access 2021.

[9] P. Suresh Kumar a, H.S. Behera, Anisha Kumari K, Janmenjoy Nayak, and Bighnaraj Naik, "Advancement from neural networks to deep learning” in Computer Science Review, vol 38, 2020.

[10] Nevena Rankovic, Dragica Rankovic, Mirjana Ivanovic and Ljubomir Lazic, "Improved Effort and Cost Estimation Model Using Artificial Neural Networks and Taguchi Method with Different Activation Functions" Entropy 2021, 23, 854, 2021.

[11] NEVENA RANKOVIC, DRAGICA RANKOVIC, MIRJANA IVANOVIC, AND LJUBOMIR LAZIC, "A New Approach to Software Effort Estimation Using Different Artificial Neural Network Architectures and Taguchi Orthogonal Arrays" in IEEE access VOLUME 9, 2021.

[12] Somya Goyal "Comparative Analysis of Machine Learning Techniques for Software Effort Estimation" in Intelligent Computing Techniques for Smart Energy Systems. Lecture Notes in Electrical Engineering, vol 862. Springer, Singapore. https://doi.org/10.1007/978-981-19-0252-9_7, 2022.

[13] Suyash Shukla and Sandeep Kumar, "Applicability of Neural Network based Models for Software Effort Estimation" in 2019 IEEE World Congress on Services (SERVICES), pp. 339-342, 2019.

[14] Md aaquib nawaz et al. "A Methodology for Software Cost Estimation Using Machine Learning Techniques" in International conference on Recent Trends in Artificial Intelligence, IOT, Smart Cities & Applications, 2020.

[15] Ming Qin, "Lattice LSTM Model for Function Point Based Software Cost Measurement" in 2019 IEEE 8th Joint International Information Technology and Artificial Intelligence Conference (ITAIC 2019), pp. 731-735 2019.

[16] Kui Zhang, Xu Wang, Jian Ren, and Chao Liu, "Efficiency Improvement Of Function Point-Based Software Size Estimation With Deep Learning Model" in IEEE Access VOLUME 4, 2016.

[17] Shaina Arora and Nidhi Mishra, "Software Cost Estimation using Single Layer Artificial Neural Network" in International Journal of Advanced Engineering Research and Science (IJAERS), Vol 4, Issue 9, pp. 22-26, 2017.

[18] Aman Ullah, Bin Wang, Jinfan Sheng, Jun Long, Muhammad Asim, and Faiza Riaz, "A Novel Technique of Software Cost Estimation Using Flower Pollination Algorithm" in 2019 International Conference on Intelligent Computing, Automation and Systems (ICICAS), pp. 654-658, 2019.

[19] KM Mahadev, G Gowrishankar, "Estimation of Effort in Software Projects using Genetic Programming" in Int. J. Eng. Res. Technol.(IJERT), Vol 9, Issue 07, pp. 1321-1325, 2020.

[20] Virro, Holger, et al. "Random forest-based modeling of stream nutrients at national level in a data-scarce region." Science of The Total Environment, 2022.

[21] Kanneganti, Alekhya. "Using Ensemble Machine Learning Methods in Estimating Software Development Effort." (2020).

[22] Morakot Choetkiertikul, Hoa Khanh Dam, Truyen Trany, Trang Phamy, Aditya Ghose and Tim Menzies, "A deep learning model for estimating story points" in IEEE Transactions on Software Engineering, Vol 45, Issue 7, pp. 637-656, 2019.

[23] Hung Phan and Ali Jannesari. 2022. Story Point Effort Estimation by Text Level Graph Neural Network. https://doi.org/10.48550/ARXIV.2203.03062

[24] Laila Cheikhi and Alain Abran, "PROMISE and ISBSG Software Engineering Data Repositories: A Survey" in 2013 Joint Conference of the 23nd International Workshop on Software Measurement (IWSM) and the Eighth International Conference on Software Process and Product Measurement (Mensura), pp. 17-24, 2013.

[25] Ünlü, Hüseyin, et al. "Software Effort Estimation Using ISBSG Dataset: Multiple Case Studies." 2021 15th Turkish National Software Engineering Symposium (UYMS). IEEE, 2021.

[26] Tim Menzies, Guenther Ruhe, A. Günes Koru, Bart Massey, Jane Hayes, Martin Shepperd and Jia Zhou. “The PROMISE Repository of empirical software engineering data,” http://promise.site.uottawa.ca/SERepository/datasets-page.html, University of Ottawa (last accessed July, 2022).

[27] Vachik S. Dave and Kamlesh Dutta, "Neural network based models for software effort estimation: a review" in Springer Science, 2012.

[28] Leonardo Villalobos-Arias, JoseGuevara-Coto, Alexandra Martínez, Marcelo Jenkins, and Christian Quesada-López "Evaluating Hyper-Parameter Tuning Using Random Search in Support Vector Machines for Software Effort Estimation" In International Conference on Predictive Models and Data Analytics in Software Engineering pp. 31–40, 2020.

[29] Somya Goyal "Effective Software Effort Estimation using Heterogenous Stacked Ensemble" in IEEE International Conference on Signal Processing, Informatics, Communication and Energy Systems (SPICES), 2022.

[30] Zakrani abdelalia, Hain Mustaphaa, Namir Abdelwahed "Investigating the use of random forest in software effort estimation." Procedia computer science 148, pp. 343-352, 2019.

[31] Sakineh Asghari Agcheh Dizaj, Farhad Soleimanian Gharehchopogh "A New Approach to Software Cost Estimation by Improving Genetic Algorithm with Bat Algorithm" in Journal of Computer & Robotics, vol 11, 2018.

[32] A. Mary Christina, V. Banumathy "Software Cost Estimation Using Neuro Fuzzy Logic Framework" in International Journal of Research in Engineering, Science and Management", vol 2, 2019.

[33] Ahmad, S.W., Bamnote, G.R. "Whale–crow optimization (WCO)-based Optimal Regression model for Software Cost Estimation" in Sādhanā 44, 94, 2019.

[34] A. A. Fadhil, R. G. H. Alsarraj and A. M. Altaie, "Software Cost Estimation Based on Dolphin Algorithm," in IEEE Access, vol. 8, pp. 75279-75287, 2020.