Journal of Advances in Science and Technology
Vol. 19, Issue No. 2, April-2022, ISSN 2230-9659

DOI: https://doi.org/10.29070/ccx1xj72

Check for updates

Analysis: The application of different Machine Learning Techniques for Software effort Estimation

Manas Prasad Rout [1] , Prof. Sabyasachi Pattnaik [2]

1. Research Scholar, F.M. University, Balasore, Odisha, India ,
2. Professor, F.M. University, Balasore, Odisha, India

Abstract: The purpose of providing estimates in software projects is to aid professionals in makingmore accurate predictions about the costs of software development, which in turn affects the efficacy ofthe software process's preparatory and operational activities. However, software developmentorganizations sometimes struggle to provide estimates that accurately reflect the true work required tocarry out the tasks associated with a software project. Despite the fact that methods for estimating laborhave been presented in the literature, doing so remains difficult. Machine learning (ML) methods haverecently been used to address this issue. With the use of ML methods, datasets containing the results ofpreviously completed projects may be used to generate more accurate estimates. In this research, wehave explored the application of machine learning methods to the problem of estimating the timerequired to develop software. The need for software projects is growing, necessitating the constantevolution of both computer software and hardware. Competition among businesses to provide highqualitygoods in a timely manner at a reasonable price has intensified as demand for software projectshas grown. The purpose of this study is to carefully examine ML models by looking at them from fourdifferent angles ML method, estimate accuracy, model comparison, and estimation context.Researchers will find this paper's review of effort estimate using machine learning methods helpful incharting the course of future work on the use of machine learning to the estimation of softwaredevelopment efforts.

Keywords: Software, Effort Estimation, Machine Learning Techniques, Software Development

---------------------------X---------------------------

INTRODUCTION

Managing a software product's whole life cycle is the goal of software development, which entails a series of software engineering tasks. Planning, development, and maintenance are the three primary stages of a software product's life cycle (Boehm, 1984). The software development process requires the establishment of a set of rules and guidelines that apply to each stage [1].

An integral aspect of any software development lifecycle model is the estimation of software effort. In this method, we ensure that you get reliable, high-quality software on schedule and within your set budget. The amount of work or money needed to create a piece of software is used to illustrate this. For many software tasks, such as gathering requirements, conducting tests, and performing maintenance, estimating software development time may be thought of as a super domain [2]. The amount of work needed to finish a piece of software at each step of the process varies depending on the software development life cycle model used. 13% to 15% of all software projects fail to accomplish the objective due of incorrect planning and overruns of schedules, which is witnessed in the software business these decades, and this makes it difficult for software engineers to estimate the efforts appropriately**[3]**.Expert-based techniques for estimating and planning have a poor success rate, thus software project teams are under growing pressure to replace them with more objective approaches, such as those based on measurement and analysis [4].

RELATED WORK

Farah Alhamdany (2022)[5] has proposedSoftware Effort Estimation (SEE) to provide an accurate estimate of the time required to develop a piece of software (in terms of person–hours or person–months). Even though there are a lot of **effort estimation**models for it, Software Effort Estimation (SEE) remains one of the trickiest challenges in creating high-quality software. The field of SEE has seen a number of model proposals. Over or underestimating the time and energy needed to complete software projects might result in the project being scrapped. As a result, this study's primary objective is to develop a performance model for assessing software effort by conducting empirical comparisons using different Machine Learning (ML) techniques. Seven datasets have been used with various ML algorithms for Effort Estimation. Software development effort estimation is evaluated on the following datasets: China, Albrecht, Maxwell, Desharnais, Kemerer, Cocomo81, and Kitchenham. Root Mean Squared Error, Mean Absolute Error, and the area under the receiver operating characteristic curve (R-Squared) were all taken into account as assessment measures. When compared to other ML techniques for software effort estimate, the LASSO approach using the China dataset performed the best in both theoretical and practical settings.

Eliane et.al.(2022) [6] suggestto infer software development time and effort from a textual description of requirements. Recent efforts in this area of Natural Language Processing (NLP) use context-less embedding models, which are typically insufficient for properly differentiating each examined phrase. Only lately have contextualized pre-trained embedding models developed, which have shown to be much more successful than context-less models in describing textual properties. To better describe textual requirements that are used to infer effort estimates by analogy, this research recommends assessing the efficacy of pre-trained embedding models. In both the context-free and contextualized methods, generic pre-trained models underwent a process of fine-tuning. A deep learning architecture is applied to the created models, and the resulting linear output is then utilised. Insight into the viability of using contextualized pre-trained embedding models to estimate software work from requirements texts alone was gained, and the findings were highly encouraging**.**

Ripu Ranjan Sinha (2021)[7] used the techniquesfor estimating the time and money needed to complete a product or piece of software are utilized. The project's deadline and budget will grow if the estimates are off, which might lead to the project's eventual collapse. In software engineering, estimating models and methods are put to use in a variety of contexts, including budgeting, risk analysis, planning, and so on. In Software Development Life Cycle (SDLC), it is crucial to accurately estimate the amount of work involved to prevent budget overruns and schedule delays. There are two main types of effort estimating methods: a) parametric and computational, b) non-parametric and ad hoc. There has been research on non-algorithmic procedures, such as those based on soft-computing techniques, as a means of getting over the restrictions imposed by algorithmic models. Parkinson, expert judgment, machine learning (ML), and pricing to win are examples of methods that aren't algorithms. In order to address the problems inherent in parametric estimate methods, ML models have been developed. Contemporary project management and development benefit from these concepts as well. Models that aren't algorithms include things like neural networks, fuzzy logic, genetic algorithms, case-based reasoning, etc. This article provides a comprehensive overview of existing machine learning (ML) approaches to software effort assessment, including their strengths and weaknesses, potential future directions of study, and current state-of-the-art.

Mamta Pandey (2020) [8] One of the most important steps in developing software is determining how much it will cost. In the context of "classical" software, certain tried-and-true approaches and techniques have been created for estimating work. The existing estimating methods for conventional desktop or online applications may not be suited for mobile app development due to the inherent differences in nature, size, and operating environment between the two. To that end, this study sets out to provide a methodology for calculating the costs of developing mobile applications. This study uses a research strategy based on sampling several mobile app characteristics from the SAMOA dataset. This data is then sent into the chosen ML methods as input vectors. Mean absolute residual (MAR) is the unit of analysis for this study. Following the results of the experiments, a framework is proposed to suggest an ML algorithm as the best fit for improved effort estimate of the project at hand. In order to resolve the dilemmas inherent in the decision-making process, this framework employs a Mamdani-type fuzzy inference technique. The results of this study will be useful for a variety of audiences, but especially for those who estimate the time and resources needed to build mobile apps.

Ahmed BaniMustafa (2018)[9] The ability to accurately predict future outcomes is essential in software engineering. Therefore, impacting the overall success of software development, by changing its cost and needed work. Software projects have risks of running late and over budget due to the error margin in Expert-Based, Analogy-Based, and algorithmic based methodologies like COCOMO (COnstructiveCOstMOdel), Function Point Analysis, and Use-Case-Points. We suggest a new approach, based on data mining of past records, to improve estimate accuracy. In order to make this forecast, the authors of this study propose employing Naive Bayes, Logistic Regression, and Random Forests, three machine learning algorithms that were previously applied to preprocessed data from the COCOMO NASA benchmark, which included 93 projects. Five-fold cross-validation was used to test the created models, and Classification Accuracy, Precision, Recall, and Area Under the Curve were used to assess their performance. COCOMO estimate was then compared to the findings of the estimation. When compared to the COCOMO model, all of the implemented methods performed well. However, a combination of Naive Bayes and Random Forests yielded the greatest results. While Naive Bayes' ROC curve and Recall score were superior to those of the other two methods, Random Forests' Confusion Matrix and Classification Accuracy and Precision scores were higher. This study verifies the usefulness of data mining for software estimation in general, and of the used approach in particular.

PROPOSED METHODOLOGY

The primary research was located via six online libraries: IEEExplor, ACMDigital Library, Science Direct, Web of Science, SCOPUS, and Google Scholar. As far as the effort estimate community is concerned, these digital libraries constitute the gold standard [10]. Figure 1 depicts a search query string that may be used to a certain database in order to get papers that address the research questions presented there.

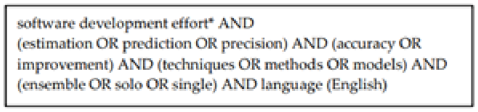

software development effort* AND
(estimation OR prediction OR precision) AND (accuracy OR
improvement) AND (techniques OR methods OR models) AND
(ensemble OR solo OR single) AND language (English)

Figure 1: Search String[10]

Inclusion criteria:

- Estimating programming time using ML methods.
- Preparing modeling data using machine learning.
- Using a hybrid model to predict development time, which uses at least two different ML methods or combines ML with a non-ML approach (such a statistics method, fuzzy set, or rough set).
- Analyses that evaluate ML models in relation to one another or to non-ML models.
- For studies that include both a conference version and a journal version, only the journal version will be included.
- Only the most recent and comprehensive publication of a given research will be included if several versions of the same study have been published.

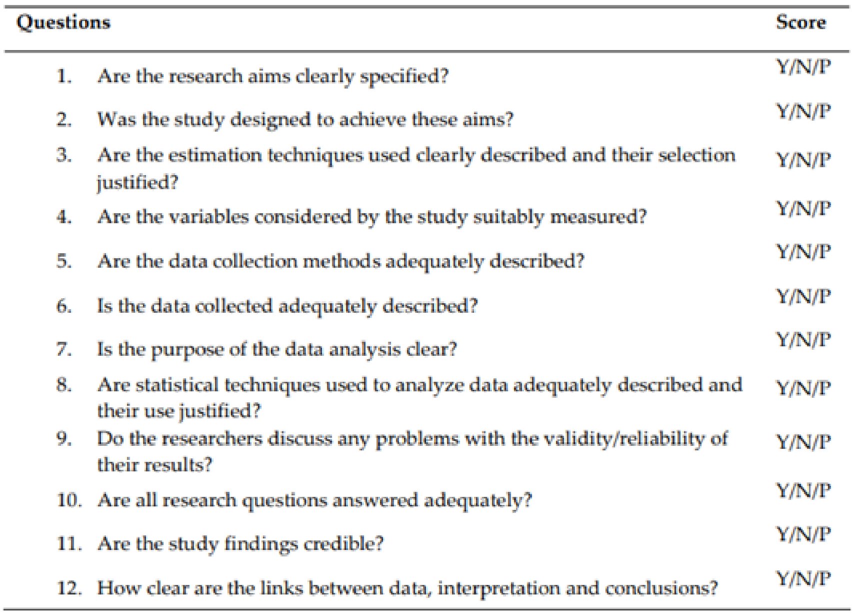
Exclusion criteria:

- To estimate software's size, timeline, or time but not its effort.
- Predicting how much time will be needed for upkeep or testing.
- Resolving problems with software project management and organization (e.g., scheduling, staff allocation, development process).
- Papers that just serve as reviews won't be considered.

Quality Assessment (QA)

We developed a battery of quality evaluation questions to evaluate the research' rigor, trustworthiness, and applicability. Table 1 displays the questions that will be asked. QA1, QA7, QA8, and QA10 are all built off of. There are just three possible responses to each question: "Y for Yes", "P for Partly", or "N for No". Here is how each of these three responses is weighted: “Yes” = 1, “Partly” = 0.5, and “No” = 0. The quality score of research may be calculated by adding up the points awarded to the various QA questions [11]. The quality of the studies was independently evaluated by two review authors. Researchers addressed their varying opinions on the quality evaluation findings and came to an accord. This review's conclusions should be trusted since we extracted and synthesized data from only studies of sufficient quality to be included in this review, namely those with quality scores above 5 (or 50% of maximum possible).

Table 1: Quality assessment questions [11]



Questions	Score
1. Are the research aims clearly specified?	Y/N/P
2. Was the study designed to achieve these aims?	Y/N/P
3. Are the estimation techniques used clearly described and their selection justified?	Y/N/P
4. Are the variables considered by the study suitably measured?	Y/N/P
5. Are the data collection methods adequately described?	Y/N/P
6. Is the data collected adequately described?	Y/N/P
7. Is the purpose of the data analysis clear?	Y/N/P
8. Are statistical techniques used to analyze data adequately described and their use justified?	Y/N/P
9. Do the researchers discuss any problems with the validity/reliability of their results?	Y/N/P
10. Are all research questions answered adequately?	Y/N/P
11. Are the study findings credible?	Y/N/P
12. How clear are the links between data, interpretation and conclusions?	Y/N/P

As a field, ML based SDEEhas 168 studies that we found. The years 1991 to 2010 cover the time frame of these articles' publication. One hundred and eighty-nine (70%) were published in journals, 48% in conference proceedings, and 1% in a book. Table 2 displays the locations at which the included studies were published. Main sources include Information and Software Technology (IST), IEEE Transactions on Software Engineering (TSE), Journal of Systems and Software (JSS), and Empirical Software Engineering (EMSE). This study draws on 88 papers, 52% of which are gathered from four highly-regarded software engineering magazines. Except for one case study, all of the investigations were experimental studies; no survey studies were included in the review. It is not guaranteed that the validation findings adequately represent the actual conditions in business, even if most of the chosen studies did employ at least one project data set from industry to evaluate ML models [12]. In reality, it is possible that the implementation of ML approaches in SDEE practice is still in its early stages due to the absence of case study and survey data from the business sector. We consider the chosen studies to be of excellent quality since our study selection technique mandated that each research's quality score be more than 5 (note that a perfect score of quality evaluation is 10). Table 3 shows that the quality of the chosen studies is generally excellent or very high, with 70% (116 of 168) falling into this category.

Table 2: Places where certain research articles can be published and made available to the public.

Publication venue	Type	# Of studies	Percent
Information and Software Technology (IST)	Journal	28	16
IEEE Transactions on Software Engineering (TSE)	Journal	24	14

Journal of Systems and Software (JSS)	Journal	18	11
Empirical Software Engineering (EMSE)	Journal	18	11
Expert Systems with Applications	Journal	8	5
International Conference on Predictive Models in Software Engineering (PROMISE)	Conference	8	5
International Software Metrics Symposium (METRICS)	Conference	8	5
International Symposium on Empirical Software Engineering and Measurement (ESEM)	Conference	6	4
International Conference on Tools with Artificial Intelligence (ICTAI)	Conference	6	4
International Conference on Software Engineering (ICSE)	Conference	4	2
Journal of Software Maintenance and Evolution: Research and Practice	Journal	4	2
Others		36	21
Total		168	100

Table 3: Studies' Quality and Relevance.

Quality level	# Of studies	Percent
Very high (8.5 ≤ score ≤ 10)	24	6.7
High (7 ≤ score ≤ 8)	92	25.7
Medium (5.5 ≤ score≤ 6.5)	52	14.5
Low (3 ≤ score ≤ 5)	140	39.1
Very low (0 ≤ score ≤ 2.5)	50	14.0
Total	358	100

TYPES OF ML TECHNIQUES USED

Out of these investigations, we are able to identify eight distinct ML methods used for software development effort estimation. Here is a rundown of all of them.Case-Based Reasoning (CBR), Artificial Neural Networks (ANN), Decision Trees (DT), Bayesian Networks (BN), Support Vector Regression (SVR), Genetic Algorithms (GA), Genetic Programming (GP), Association Rules (AR). CBR, ANN, and DT are the three most often utilized ML approaches; combined, they were used by 80% of the chosen studies (Fig. 3). The specifics of how each research made use of ML methods [13]. Only the total amount of attention paid to research on each ML approach during the previous two decades is shown in Fig. 3.

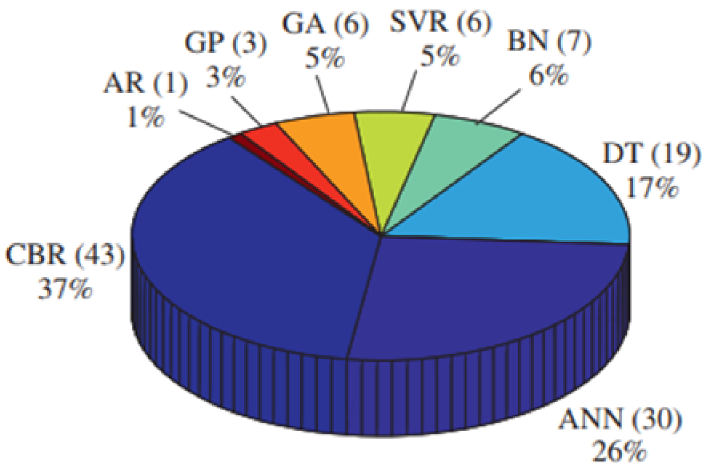


Figure 3: Distribution of research projects across various machine learning methods.

PRECISION OF ML MODEL ESTIMATES

Due to the fact that ML models are data-driven, previous project information is crucial for both the model building and validation phases. It is important to consider both the validation technique used and the historical project data set used to build and test the ML model's estimate accuracy.All of the machine learning models discussed here were built and tested using a wide range of archived project data. The most commonly utilized data sets combined with their related metadata are provided in Table 4. Data set type (inside or across firm), number of projects in the data set, data set source, and proportion of studies using the data set are all pieces of information that are pertinent to the question at hand. In terms of validation strategies, the chosen studies mostly use Holdout, Leave-One-Out Cross-Validation (LOOCV), and n-fold Cross-Validation ($n > 1$) [14]. To be more precise, 32 studies (38%), 31 studies (37%), and 16 studies (19%) respectively employed Holdout, LOOCV, and n-fold Cross-Validation as their validation techniques of choice.

When assessing ML models, it is important to take into account not just the data set and validation approach, but also the accuracy measure. The precision of an effort estimate may be evaluated using a number of metrics, each of which measures accuracy in a somewhat different way. That's why it's so important to use the right criteria for measuring precision when assessing the precision of estimates. It is revealed in the chosen research that MMRE (Mean Magnitude of Relative Error), Pred (Percentage of predictions that are within 25% of the actual value), and MdMRE (Median Magnitude of Relative Error) are the three most common accuracy measures. Specifically, the numbers (percentages) of the studies that employed these three indicators are 75 (89%) for MMRE, 55 (65%) for Pred, and 31 (37%) for MdMRE. In this evaluation, we used the widely-used MMRE, Pred, and MdMRE metrics to measure the estimate accuracy of ML models.Box plots (Fig. 5a-c) show the distributions of MMRE, Pred, and MdMRE values of ML models, which we utilized to visualize the estimate accuracy. We didn't include MMRE or Pred for AR in the relevant statistics since there weren't enough observations to make a meaningful box plot. It is for this reason that Fig. 5c does not include box plots of MdMRE for SVR, GP, and AR.

A better estimate is denoted by a smaller MMRE and MdMRE and a larger Pred. In terms of MMRE and Pred (see Fig. 5a and b), ANN and SVR perform best (median MMRE around 35% and median Pred around 70%), followed by CBR, DT, and GP (median MMRE and median Pred around 50% for each), while BN performs worst (median MMRE around 100% and median Pred around 30%). In terms of ML model performance as evaluated by MdMRE (see Fig. 5c), CBR and ANN both have median MdMREs close to 30%, whereas BN and DT both have median MdMREs close to 45%. In addition to what has already been mentioned, we can see from Fig. 5a that the median MMRE values for CBR, ANN, DT, BN, and GP are all very close to the center of their respective boxes, indicating that the MMRE values for these ML models are distributed symmetrically around their medians. In contrast, the distribution of MMRE values for SVR exhibits a slight negative skewness. Figure 5a further demonstrates that the MMRE values for CBR, ANN, SVR, and GP are more consistent than those for DT and BN, thanks to their smaller standard deviations and shorter confidence intervals.

Intuitively grasping the estimate accuracy of ML models is made possible with the help of Fig. 5's box plots.

Table 4: Examples of Data Sets Used in the Development and Testing of Machine Learning Models

Data set	Type	# Of studies	Percent	# Of projects
Deshamais	W	24	29	81
COCOMO	C	19	23	63
ISBSG	C	17	20	>1000[a]
Albrecht	W	12	14	24
Kemerer	W	11	13	15
NASA	W	9	11	18
Tukutuku	C	7	8	>100[a]

W = within-company, C = cross-company

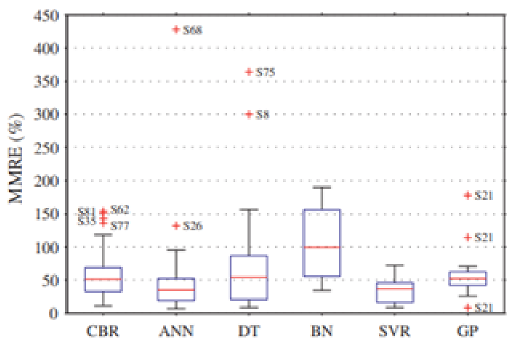


Figure 5: a) Plots of MMRE

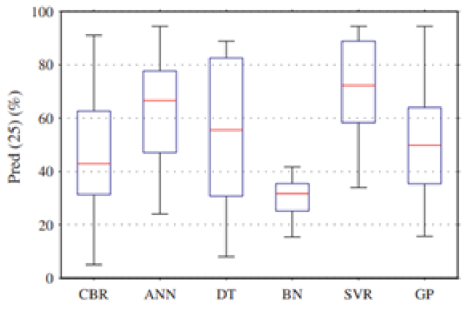


Figure 5: b) Plots of Pred

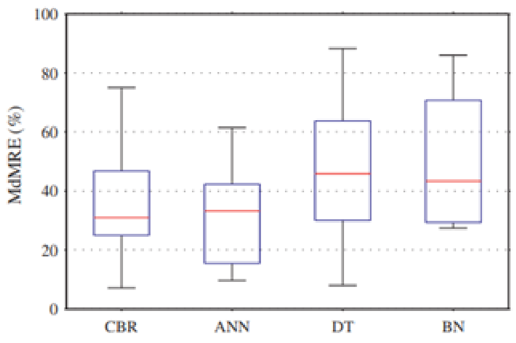


Figure 5: c) Plots of MdMRE

We focus on the 4.25 MAE value and 0.17 standard deviation achieved by using the contextualized pre-trained model BERT with fine-tuning in a single repository including many projects. When compared to other works, this reflects a highly good outcome. The suggested estimate approach has the potential for generalization, is fast, and requires few computer resources. The fine-tuning procedure offers this benefit by reducing the amount of work needed to fulfill either new or current needs.

VALIDITY THREATS

This section explains threats to the validity of this paper regarding internal, external and construct validity.

- To correctly extract the data from the selected studies, five authors of this study read each paper independently. The data extracted for each paper were compared. All disagreements on data quality were discussed among all authors, and the consensus was eventually reached. The only best estimation accuracy obtained from the optimal configuration of the ensemble and solo techniques was extracted. However, there might still be a bias in data extraction; nevertheless, using the optimal configuration is a good way to minimize the bias.
- The accuracy values of the estimation are extracted from papers focusing on solo and ensemble effort estimation (EEE) models.
- In addition, these values are derived from various experimental designs involving the selection of techniques, data sets, data preparation, and employing the method used to validate the model. Note that all of these steps directly affect the accuracy of estimations.
- It is nonetheless believed that the estimation accuracy results obtained from various experimental designs are more robust than those from a uniform design.

This study used publicly and non-publicly domain datasets. The projects are collected from different countries and organizations, and their features are diverse. This makes them adequate for evaluating the accuracy of solo and ensemble effort estimation techniques. However, it will be beneficial to replicate this study using other machine learning techniques and datasets with similar characteristics [15]. The purpose of construct validity is to answer the question of the reliability of the performances measured through this study. Since this study focuses only on the accuracy of effort estimates, two evaluation criteria [MMRE (Mean Magnitude Relative Error), PRED]are used. The main reasons behind this choice are that these performance criteria are unbiased, less fragile, and commonly used in asymmetry assumptions. Additionally, there are several other metrics that are less vulnerable to asymmetry assumptions such as Mean Inverted Balanced Relative Error (MIBRE) and Mean Balanced Relative Error (MBRE) that can also be used to assess the proposed technique.

CONCLUSION

As a result of the work presented in this article, we have advanced the field of software project estimating. For this study, we zeroed emphasis on the ensemble and solo methods of effort estimate that are based on machine learning. Our participation in this study is twofold. To begin, we investigated the current state of the art in the field of effort estimating, both in terms of ensemble and individual methods. By adhering to the standard procedure for a Systematic Literature Review (SLR) in the field of software engineering, we were able to successfully solicit primary research. Second, we used widely-used accuracy performance measures (MMRE and PRED) to PD and NPD datasets and compared and assessed both methods. Our study may serve as a springboard for more research in this area. Our study findings will help developers choose an appropriate effort estimate method for their future software projects. Case-based reasoning (CBR), ANN, DT, BN, SVR, GA, GP, and association rules are only some of the ML methods that have been implemented in SDEE (AR). The most popular ones are CBR, ANN, and DT. Most ML models have estimation accuracy that is nearly acceptable. The average MMRE for ML models is about 35%, the average PredMRE is around 45%, and the average MdMRE is around 50%. In addition, the average MMRE for ANN is about 35%, whereas for CBR and DT it is approximately 50% and 55%, respectively. Studies consistently show that ML models outperform non-ML models in terms of accuracy. The most common alternative to ML models is the regression model.

References

1. Boehm BW. 1984. Software engineering economics. IEEE Transactions on Software Engineering 10(1):4–21 DOI 10.1109/TSE.1984.5010193.

2. Choudhary, K. (2010). GA Based Optimization of Software Development Effort Estimation. International Journal Of Computer Science And Technology, Vol.(1), pg. 38–40.

3. Dan, Z. (2013). Improving the accuracy in software effort estimation: Using artificial neural network model based on particle swarm optimization. In Proceedings of 2013 IEEE International Conference on Service Operations and Logistics, and Informatics, SOLI 2013(pp. 180–185).

4. Garbajosa, J. (2008). The emerging ISO International Standard for Certification of Software Engineering Professionals. In IFIP International Federation for Information Processing (Vol. 280, pp. 173–178).

5. Alhamdany, farah, and laheeb Ibrahim. “Software Development Effort Estimation Techniques: A Survey.” مجلةالتربيةوالعلم, vol. 31, no. 1, Mar. 2022, p. 80, https://doi.org/10.33899/edusj.2022.132274.1201.

6. Eliane Maria De BortoliFávero, Dalcimar Casanova, Andrey Ricardo Pimentel, “SE3M: A model for software effort estimation using pre-trained embedding models,” Information and Software Technology, Volume 147, 2022, 106886, ISSN 0950-5849, https://doi.org/10.1016/j.infsof.2022.106886.

7. Sinha, Ripu Ranjan & Gora, Rajani. (2021). Software Effort Estimation Using Machine Learning Techniques. 10.1007/978-981-15-5421-6_8.

8. Mamta Pandey, RatneshLitoriya, and Prateek Pandey, “International Journal of Software Engineering and Knowledge Engineering” 2020 30:01, 23-41

9. A.BaniMustafa, "Predicting Software Effort Estimation Using Machine Learning Techniques,"*2018 8th International Conference on Computer Science and Information Technology (CSIT)*, 2018, pp. 249-256, doi: 10.1109/CSIT.2018.8486222.

10. Maleki, I., Ghaffari, A., &Masdari, M. (2014). A New Approach for Software Cost Estimation with Hybrid Genetic Algorithm and Ant Colony Optimization. International Journal of Innovation and Applied Studies, 5(1), 72–81

11. Missier, P., Lalk, G., Verykios, V., Grillo, F., Lorusso, T., &Angeletti, P. (2003). Improving data quality in practice: A case study in the italian public administration. Distributed and Parallel Databases, 13(2), 135– 160.

12. Rajper*, S., & and Zubair A. Shaikh. (2016). Software Development Cost Estimation Approaches - A Survey. Indian Journal of Science and Technology, 9((31)), pg.1–5.

13. Sandhu, G. S. (2014). A Bayesian Network Model of the Particle Swarm Optimization for Software Effort Estimation tool. International Journal of Computer Applications, 96(4), 52–58.

14. Vu Nguyen, Bert Steece, B. B., Nguyen, V., Steece, B., Boehm, B., & Vu Nguyen, Bert Steece, B. B. (2008). A Constrained Regression Technique for COCOMO Calibration. Acm 978-1-59593-971-5/08/10, 1– 10.

15. Borade, J. G., &Khalkar, V. R. (2013). Software Project Effort and Cost Estimation Techniques. International Journal of Advanced Research in Computer Science and Software Engineering, 3(8), pg. 730–739.