A Machine Learning Framework using Artificial Neural Networks (ANN) for Software Development Prediction

Nupur Parihar
Assistant Professor, Swami Vivekananda College of Engineering (SVCE) Indore
nupurparihar68@gmail.com

Abstract- *To get reliable software on time and under budget, you need to be able to accurately forecast how much work will go into developing it. There have been a number of software effort estimating models made in the last few years, but it's still hard to get reliable estimates for software projects. Software failures are primarily attributed to inaccurate effort estimation during initial phases. So, a lot of researchers are utilizing AI to make new models and make the ones that already exist better. Dynamic circumstances in software development technology complicate the predictability of work estimation. This article talks about the most popular ways to estimate how much work software will take. ANN (Artificial Neural Network) can simulate a complicated set of relationships between the dependent variable (effort) and the independent factors (cost drivers), which makes it a promising tool for estimate. This study provides a review grounded in the performance analysis of artificial neural networks (ANN) and juxtaposes the outcomes of linear regression models in effort estimation.*

Keywords- software development, effort prediction, linear regression, Artificial Neural Network

I. INTRODUCTION

Software managers need to maintain a close eye on the quality and reliability of the software and be ready to change how they use their resources quickly in order to develop dependable software on time and within budget. Managing resources well depends a lot on judging the quality and reliability of the software being developed, as well as knowing the quality and cost of the resources that are available. The latter is usually understood qualitatively and is often harder to express in quantitative terms. But it is hard for managers to measure quality and reliability when a product is still being made. Indicative measurements of software reliability and quality assist in forecasting resource needs to fulfill stated performance criteria. One of the most significant things to do to reach the aforementioned goal is to predict software development activities. If you don't accurately guess how much work will be needed, you could end up spending more money than you planned or getting software that doesn't work as well as it should. Overestimating the amount of work needed wastes software development resources, while underestimating the amount of work needed causes delays in the schedule, poor product quality, and fines.

II. RELATED WORK

Mendes et al. [1] categorized software effort estimating techniques into three types: expert judgment-based models, algorithmic models, and artificial intelligence (AI)-based models. Expert judgment-based models gauge efforts by leveraging the insights of specialists involved in the software development process. The disadvantages include standardization and the analytical reasoning of specialists [2]. In the software development process, several indicators, including size, effort, and cost drivers, function as independent factors, while effort and cost serve as dependent variables. The parametric models, such as Constructive Cost Model (COCOMO) [3], function points (FPs) [4], Software Lifecycle Management (SLIM) [5], and queuing networks or Petri net [6], are limited by traditional statistical distribution properties. Artificial neural networks (ANN), genetic algorithms (GA), regression trees, case-based reasoning models, neuro-fuzzy, Bayesian networks, and fuzzy logic are all used by AI-based models to figure out how much time and money it will take [7]. These models that are based on data use data from past software projects that are comparable to train. The trained model forecasts development endeavors for new projects [8]. ANN has been utilized for forecasting the aggregate number of failures within a specified timeframe, the interval between failures, and modules susceptible to faults [9]. It has been determined that ANNs provide superior prediction capabilities compared to the majority of traditional approaches [10]. Neural networks are naturally parallel, and they have a lot of potential since they can learn how to model non-linear interactions.

A. Assessment of models' accuracy

The primary issue of an effort estimating model lies in its ability to generate precise forecasts. The mean magnitude relative error (MMRE) and the percentage relative error deviation within x (PRED(x)) are the two most frequent accuracy predictive statistics [11]. The value of magnitude relative error (MRE) is what both of these measures are based on. MRE is a standardized way to measure how different the actual data values (in this case, effort values) are from the estimated values [20].

MRE i = absolute (Actual Effort i–Predicted Effort i) / Actual Effort i … (1)

The mean MRE (MMRE) is the average value of this indicator over all observations in the sample. From a project manager's point of view, a lower MMRE number usually means a more accurate model. The pred measure shows how well a set of data points fits together, depending on the MRE values for each point:

pred (*l*) = i/n … (2)

In equation (2), l is the chosen threshold value for MRE (from equation (1)), i is the number of data points that have MRE less than or equal to l, and n is the total number of data points. For example, if pred(0.20) = 30%, we may state that 30% of the fitted values are within 20% of their real values. Dunsmore et al. [12] assert that values of MMRE $\leq$ 25% and PRED(25) > 75% are optimal for a precise effort model.

B. Linear regression

Linear Regression is a type of computational technique. Leung and Fan [13] assert that it is an empirical model necessitating historical data from previous initiatives to assess current projects. Boehm et al. [14] and Singh et al. [15] classify LR as a category of effort estimation strategies employed to ascertain the relationship between the dependent variable (Y) and the independent variables (Xi) [16]. What is an MLR model?

$$Y = \beta 0 + \beta 1X1 + \beta 1X1 + ... + \beta nXn + \varepsilon \quad \dots (3)$$

where X1, X2, ..., Xn are regressors; β0 is the intercept parameter; β1, β2, ..., βn are the regression coefficients; and ε is the error component. The coefficient of determination R^2 is used to see how good the model is [17]. It tells you how much of the entire variability of the independent variable around the mean is due to the regression model fitting [18]. When you require a simple model and analytic tool for estimating effort to help with the first tries, you can use regression [19]. The use of the LR approach necessitates the validation of its underlying assumptions. The main things to think about are Linearity: The relationship between each Xi and Y is linear, therefore the model does a good job of describing how the data behaves; *ƒ* The error component is a variable that is independent and normally distributed, with a mean value of zero and a constant variance. If there are problems with the data collection or the wrong model is used, these assumptions may not be met. There are a number of ways to choose the independent variables that will be used in the LR model. One method is to include all the independent variables that are thought to be important, while others employ stepwise methods like forward regression, backward regression, and stepwise regression. The stepwise model is the most common one used in this study. This strategy incorporates the independent variables sequentially (Xi), commencing with the variable exhibiting the highest correlation with Y. At each step, the R^2 value is checked to see if each of the variables that were already included is helping to raise it. If not, it is left out.

C. Artificial neural networks

Artificial neural networks (ANNs) are huge parallel systems that are based on the structure of biological neural networks. They are made up of simple, connected elements called artificial neurons. If the weighted sum of its inputs is greater than a specific threshold, the neuron sends out an output. This output then becomes an input to other neurons in the network that can either excite or inhibit them. The procedure goes on until one or more outputs are made. The arrangement of units in layers that are connected to each other makes up the neural network.

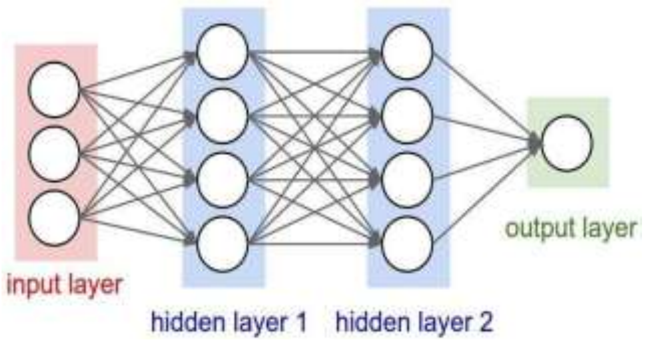


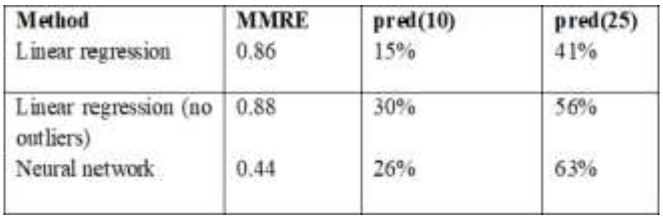
Fig1: ANN

The architectures that come out of this learn about the data that is connected to the problem in order to solve it. There are a lot of various ways to learn. Forward feed The most frequent type of ANN is the multilayer perceptron. The error back propagation approach, which needs a differentiable activation function, is what most people use to train multi-layer structures.

D. Comparison of Techniques

Two-thirds of the 54 observations were used to train Multi-Layer Perceptron (MLP) networks, and one-third were used to test them [20]. Training ceased with the minimization of the testing error, and the minimal testing error was utilized to determine the specific network architecture for performance evaluation.

Table 1: Overall result



Method	MMRE	pred(10)	pred(25)
Linear regression	0.86	15%	41%
Linear regression (no outliers)	0.88	30%	56%
Neural network	0.44	26%	63%

The results presented in Table 2 indicate that the neural network model outperforms other models in terms of Mean Magnitude of Relative Error (MMRE). This superior performance is largely attributed to the model's ability to capture nonlinear relationships and interactions within the data, which are often overlooked or inadequately addressed by traditional regression-based approaches. Regression models typically struggle to account for complex interdependencies when the dataset is relatively small, limiting their predictive accuracy. In contrast, neural networks can effectively model these intricacies, resulting in more precise software development effort estimations and making them a preferred choice for projects with complex attribute interactions.

III. CONCLUSION AND FUTURE WORK

Estimating software development effort accurately remains a significant challenge in software engineering. Over the years, researchers have developed various models to predict the effort required for software projects, including algorithmic models, regression models, and heuristic approaches. However, none of these models consistently deliver highly accurate predictions, especially when evaluated using standard metrics such as Mean Magnitude of Relative Error (MMRE). In this context, Artificial Neural Networks (ANNs) have emerged as a competitive alternative to traditional estimation techniques. ANNs leverage their ability to model complex, non-linear relationships between project attributes and effort requirements, providing more adaptable and flexible predictions. Despite their promise, ANN models are not universally optimal and often require careful tuning and feature selection to achieve acceptable performance.

Given these limitations, our research aims to extend the study by exploring other machine learning approaches, including genetic algorithms and hybrid models that combine multiple techniques to enhance prediction accuracy. By integrating complementary methods, it is possible to capture different patterns in project data and generate more reliable effort estimates. The objective of this replication study is to identify the most effective combination of algorithms and establish a robust framework for software development effort estimation, ultimately assisting project managers in planning and resource allocation more efficiently.

REFERENCES

[1] A. Srilatha and P. Praveen, "Deep Learning for Farmland Assessment and Developing an Automatic Crop Recommendation System Using GCN," 2024 International Conference on IoT Based Control Networks and Intelligent Systems (ICICNIS), Bengaluru, India, 2024, pp. 1735-1741,
[2] Boehm, B.W.: 'Software engineering economics' (Prentice-Hall, Englewood Cliffs, NJ, 1981)
[3] Praveen Pappula, "A Novel Binary Search Tree Method to Find an Item Using Scaling", The International Arab Journal of Information Technology (IAJIT) ,Volume 19, Number 05, pp. 122 - 129, September 2022, doi: 10.34028/iajit/19/5/2 .17. D. C. Montgomery, E. A. Peck and G. G. Vining, Introduction to Linear Regression Analysis 4th Edition. John Wiley & Sons, Inc., Hoboken NJ, 2006.
[4] Mohammed Ali, P. Praveen, Sampath Kumar, Sallauddin Mohmmad, M. Sruthi, "A survey report on cloud based cryptography and steganography procedures", International Conference on Research in Sciences, p9Engineering, and Technology, AIP Conf. Proc. 2971, 020040-1–020040-8; https://doi.org/10.1063/5.0196050
[5] Jorgensen, M., Shepperd, M.: 'A systematic review of software development cost estimation studies', IEEE Trans. Softw. Eng., 2007, 33, (1), pp. 33–53
[6] Sampath Kumar Tallapally, Mohammed Ali Shaik, P. Praveen, Masani Ruchi Nandhan, "Information security in cloud utilizing AES", AIP Conference Proceedings, 2971, 020050 (2024), https://doi.org/10.1063/5.0196056
[7] P. Praveen, K. Srilatha, M. Sathvika, E. Nishitha and M. Nikhil, "Prediction of Alzheimer's Disease using Deep Learning Algorithms," *2023 2nd International Conference on Applied Artificial Intelligence and Computing (ICAAIC)*, Salem, India, 2023, pp. 587-594, doi: 10.1109/ICAAIC56838.2023.10140746.10. Berry, M.J.A., Linoff, G.: 'Data mining techniques for marketing sales, and customer support' (Wiley, New York, 1997)
[8] D. Port, V. Nguyen and T. Menzies, Studies of Confidence in Software Cost Estimation Research Based on the Criterions MMRE and PRED, http://menzies.us/pdf/09predmmre.pdf, 2009.
[9] Wittig and Finnie, 1994], G.E. Wittig and G.R. Finnie, Using artificial neural networks and function points to estimate 4GL software development effort, Australian Journal of Information Systems 1(2), 87-94, 1994.