European Journal of Clinical Pharmacy
Print ISSN: 2385-409X **Online ISSN**: 3105-0409
Website: https://farmclin.com/

Original Research Article

Using an Extreme Learning Machine to Estimate Software Development Efforts

Name of Author:
Chetana Pareta[1], Dr. Sunil Gupta[2]

Affiliation: *[1]Ph.D. Scholar*
School of CSE
Jaipur National University, Jaipur, India
[2] Professor & Head, School of Computer & Systems Sciences,
Jaipur National University, Jaipur, India

Received: 11-11-2025
Revised: 26-11-2025
Accepted: 10-12-2025
Published: 31-12-2025

Abstract: Estimating the quantity of work required for software development remains a challenge for project managers in the software industry. Several new techniques have been put forth to improve these approximations' accuracy. Since there are many different methods that have been published in the literature, evaluating accuracy is crucial. In this work, we provide a novel method that we name Real-Time Extreme Learning Machine (RT-ELM), which is based on an online sequential learning algorithm. The extreme learning machine can now adapt to the creation of new projects within a software organization thanks to the modification of this algorithm, which makes continuous learning possible. RT-ELM's efficacy is compared to that of traditional training and testing techniques. Additionally, the experiments employed the radial basis function (RBF) and additive hidden nodes. The results show that RT-ELM with continuous learning outperforms conventional methods. Additionally, it was shown that data had an impact on RBF and additive hidden node efficiency. Data from commercial and research settings were used to validate the findings. In the context of software engineering, accurate and reliable effort estimation is a crucial part of the project management process that helps managers maintain project control. The current study is to: (i) identify critical elements influencing effort estimation by correlation analysis; and (ii) apply the Extreme Learning Machine (ELM) model for effort estimation, comparing its performance to previous models in academic literature. The goal of the study was to determine which method produces the most accurate effort prediction. The models were compared in terms of anticipated precision using statistical tests and the mean absolute residue (MAR) criterion. The ELM model exceeds other models in terms of accuracy when it comes to software design effort prediction, according to the results, which also highlighted important variables for effort estimation. These are a few of the main conclusions. Project success is therefore increased when machine learning techniques are applied to effort estimation in order to improve the accuracy of time and expense estimations.

Keywords: Radial basis function, Real Time, Extreme Learning Machine, Software Development Efforts

INTRODUCTION

In recent months, data mining and machine learning methods have drawn in an attempt to increase estimation accuracy. However, because different metrics are used to determine accuracy, it is challenging to compare different methods. Accuracy is also impacted by the raw data and criteria employed [1]. Task with analogous past endeavours in order to calculate effort. Expert estimating generates effort estimates based on expert assessments. Model-based approaches establish relationships by utilizing historical data. One of the most often used AI techniques for estimating effort is neural networks. For this study, simple method, the (RT-ELM), with a single experimentally chosen parameter. To satisfy the needs of the software industry, methods for predicting software development efforts have been studied for a long time. Estimating software development effort, which is an essential component of project management [2]. Reducing risk and increasing project success rates need accurate effort evaluation. The software project management process involves the use of certain strategies and procedures to achieve project

objectives. In order to meet client expectations, the development process estimates both quantitative and qualitative parameters. As Saraiva notes, efficient software measuring tools help with decision-making and enable effective data interpretation. For projects to be finished on time and within budget, organizations need to choose the best effort estimating method from a range of possibilities [3]. One often used technique is Expert Judgment, where project managers rely on the expertise of experts; however, this approach has a disadvantage in that it is prone to human error. For developers and project managers, estimating software effort remains a major challenge due to the varying work required at different stages of the project lifecycle. Traditional methods need careful activity documentation and can be challenging and time-consuming. Additionally, there are a lot of variables that can't always be predicted, such as the engineers' expertise, the team's project history, and many others. a potential remedy for the problem of (SDEE) [4]. ML techniques can learn and adjust on their own over time to make better predictions. Additionally, by employing data-driven analysis, these methods simplify decision-making and minimize the need for human intervention. By using machine learning techniques, experts can spend less time estimating project tasks and more time focusing on other system responsibilities that satisfy customer requirements. Project objectives are not always well stated at the outset, which leads to ambiguity in the estimations needed for project development and affects effective project management. Accurate resource allocation requires realistic effort estimates [5].

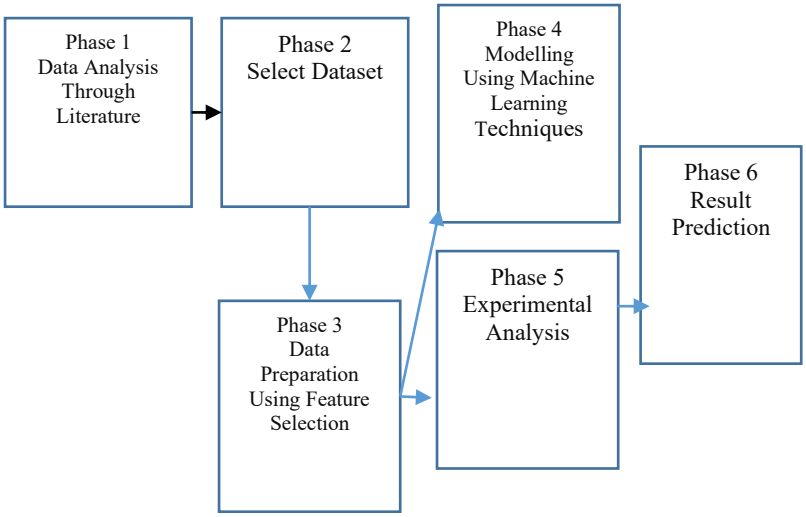
Recent years have seen the introduction of several modified ELM algorithms, which have found application in a wide range of fields, such as deep learning, device localization using spatiotemporal data, handling Gaussian and non-Gaussian noise issues, and estimating gas utilization in blast furnaces. However, in the systematic literature search, only one study specifically applied the ELM technique for (SDEE). The feedforward network is the most popular kind of (ANN) [8]. It was noted by Huang et al. that training an ELM can be completed faster and with better generalization performance than backpropagation training of a neural network. We also investigated the accuracy of the software effort estimation and the best attributes to select for task estimation. This prompts the following research questions:

2. RQ1: What qualities are essential for obtaining more accurate software development task estimates?

3. RQ2: When estimating effort, which machine learning technique produces the best accuracy?

We examined the relevance that are on par with or better than those of other models. Among the many advantages of using Enterprise Life Management (ELM) in software engineering are time and cost savings across the course of the project.

2. RELATED WORK



Oliveira et al. [25] employed a (GA) as ML, and [13] created a feature model for software task estimation based on GA. Both studies show that (SDEE) is still important despite a large body of research from academia and industry worldwide. The input data, algorithms, and accuracy evaluation standards are the main challenges, as they must all be considered concurrently to reach a well-informed judgment.

Fig. 1. Suggested Approach for Precise Effort Forecasting

Boehm et al. [1] argue that rather than relying just on one strategy for SDEE, multiple approaches should be compared to enable better decision-making. Project size is one of the most crucial input elements that determine how much effort is put into software development. The Extreme Learning Machine (ELM) has gained prominence due to its advantages over conventional feedforward backpropagation neural networks. ELM has superior generalization abilities in addition to being extraordinarily fast. ELM's straightforward and simple design encouraged the authors to utilize it for software development effort estimation. ELM has also been used to forecast maintainability in object-oriented systems [11].

In these contexts, software development projects are often bid on and produced concurrently, necessitating effort estimation early in the project lifecycle. Estimating effort is essential at the start of the project. Since the true effort is known by the end, the model can be adjusted. In this case, online

sequential estimation is most effective. Consequently, the study employed the (OS-ELM) [7], which integrates training and testing stages. Its efficacy was compared to that of traditional training and evaluation methods. Additionally, the study examined the impacts of radial basis functions and additive hidden nodes, both of which needed an equal number of parameters to implement. In the context of optimizing input resources, [27] looked into the use of the Bees In another study, [26] highlighted the superiority of evolutionary algorithms and proposed estimating software development labor using non-algorithmic methods. Minku's group conducted experimental research on automated machine learning ensembles [29], and the findings showed good performance on a range of datasets.

3. RESEARCH METHODOLOGY

The process for creating the (ELM) model for is described in this section. The suggested structure for precise effort prediction is shown in Figure 1. The two primary phases of (SDEE) are usually model creation and model evaluation, often known as training and testing. The model is constructed using a subset of the dataset; the remainder is set aside for testing. A model's capacity to generalize to previously encountered data may be restricted, even though it may function well on training data. The training data is frequently divided into two sets—one for training and another for validation, when model parameters are adjusted—to address this problem. When submitting bids for projects, managers in the software business must calculate the cost of development. For software firms, proper effort assessment is essential because project costs are directly related to human labor. Usually, estimates of program size and other environmental parameters are used to make this calculation. (RT-ELM) data used for testing and training. Rather, as new data becomes available, it concentrates on continuously learning and confirming its prediction capabilities. Using the data at hand, the estimating process seeks to create a link between the dependent and independent variables.

3.1. Data Set

Function Points are used to measure software size. There are 77 data points total; 60 are used for testing and 17 for training. Both size and effort are expressed in hours and function points, respectively, in the Maxwell dataset. There are 62 projects in this dataset; 40 are used for testing, and 22 are for training. One independent variable (size) and one dependent variable (effort) were employed in the study. There are 231 data points in the Lopez dataset; 163 are utilized for training and 68 are for testing. Two attributes are included in this dataset: reused code and new and modified code. While effort is measured in minutes, code size is measured in lines of code. The ISBSG dataset includes 532 excellent projects that were created in industrial settings across many nations. These projects are tiny in size and were developed in an academic setting. Of them, 182 are used for testing, while 350 are utilized for training. There are nine independent variables in this dataset; the size is expressed in function points and the effort is expressed in hours.

3.2. Data Set Description

As seen in Table 1, these attributes are categorized as either numeric or category. Since the "Project ID" feature had no bearing on the project effort estimate, only numerical data were chosen for this analysis. The factors that were independent and dependent were then determined. The effort needed to finish a project is the primary topic of software development effort estimation methods. Consequently, the linked qualities "Team Experience," "Manager Experience," "Year End," "Length," "Transactions," "Entities," "Points Non-Adjusted," "Envergure," and "Points Adjusted" were chosen as independent variables, while "Effort" was designated as the dependent variable. The statistical metrics for the independent and dependent qualities is shown in Table 2. The 81 (length) of 11.7 months, ranging from 1 to 39 months. There is not much of a difference between the two variables that measure program size, "Points Non-Adjusted" and "Points Adjusted," with the average of Points Non-Adjusted being 304 and Points Adjusted being 289the recorded effort ranged from 546 to 23,940 person-hours. The distribution of effort, which is the dependent variable and is measured in person-hours, is shown by the histogram in Figure 3. With a few extremely high outliers and the majority of records concentrated at lower levels, the data is favourably biased.

3.3. Real Time-Extreme Learning Machine (RT-ELM)

Due to their ability to learn complex functions, Artificial Neural Networks (ANNs) are widely used

- The potential for local minima to be reached.
- Establishing the stopping criteria.
- Iterative learning takes a lot of time to determine weights.
- Choosing the optimal number of layers to attain the necessary degree of accuracy.

One hidden layer with LLL nodes makes up the ELM. The output function for NNN unique samples (xi,ti)(x_i, t_i)(xi,ti).

The OS-ELM is appropriate for software

completion or project bidding due to its commencement and sequential learning phases. After learning, the model's parameters are set and applied to future projections. However, in the lack of a clear testing phase, the estimating model must constantly adjust to changing software development environments and processes.

Comprehensive derivations of OS-ELM can be found in the literature. This is a synopsis of the procedure:

- Assign Weights and Biases Randomly: Random input weights wiw_iwi and biases bib_ibi are allocated to additive (ADD) or radial basis function (RBF) hidden nodes.
- The architecture of ELM improves learning effectiveness and efficiency while continuously adapting to new information.

Since RT-ELM is an extension of OS-ELM for continuous learning, it is especially well-suited for the software sector. can be included in the network. For both types, the effect width for RBF nodes must be positive, even though the number of randomly chosen parameters is the same. These randomly selected variables are set during the initialization process and may affect the network's performance.

Figure 2 shows an illustration of the RT-ELM flow chart. The output vector β\betaβ for each input can be changed freely with RT-ELM, in contrast to conventional techniques that employ predetermined parameters for training and testing. Tables 2 through 9 display the statistical properties of the errors for four distinct datasets with both constant and variable β\betaβ for RBF and ADD nodes. Similar patterns can be seen in the standard deviations and interquartile ranges, and as the RMSE falls, the correlation between the observed and projected data gets better. Despite the small decrease in the RMSE, there is a slight loss in correlation with Lopez data.

Out of all of the aforementioned, the newly created ELM completely avoids iterative learning [4, 5, 6]. Contrary to popular belief, ELM allows the random selection of This approach has outstanding generalization capabilities in addition to being quick. The use of ELM in numerous applications has been transformed by this non-iterative approach. The authors were persuaded to select ELM for SDEE by these benefits. Project bidding or projects where β = [β1, βL] T between the output node and the hidden layer of L nodes in the software industry. Either piece by piece or chunk by chunk, completion occurs. Thus, the OS-ELM is appropriate for this use case. stage of learning in stages. software development techniques change throughout time. Changes occur in the development environment. Additionally, the programmers' output differs. Therefore, the estimating model needs to be modified to account for the evolving circumstances. As a result, there is no testing step in the learning process. This is a synopsis of OS-ELM; [7] contains the full derivation.

• Step 1: Phase of Initialization: One item of data is used to initialize the ELM. Determine the value of H, the neural network's hidden layer output matrix. In [6], it is demonstrated that L≤ N hidden neurons are necessary if the activation function is infinitely differentiable. H stays fixed when the hidden node weights and biases are set and β needs to be computed. The lowest norm least squares solution for the buried neurons (L) in equation (4). We only need to make an empirical determination for this parameter. The initial set (N0) should be at least as large as (L). g(wixN0+ b1) g(wLxN0+ bL) [g(wixj+ b1) g(wixj+ bL)] = H0H0 is a N0 X L matrix. Determine the output's initial weight. β̂ = H† T, T β(0) = P H TP0To .The procedure can be checked for the entire collection of data using Figure 1 depicts the ELM architecture, with To=[t1,…..,tN0] and P0=(H T H0)-1. Each t stands for the actual work being done on the project. Put k at zero.[g(w1,b1,xk+1)…… g(wL,bL,xk+1)] = hk+1, where k is the processing moment. Calculate how much work will be required to complete the project. Pk minus Pkhk+1hTk+1 is equal to Pk+1 Pk(1+ hTk+1Pk hk+1) Pkhk+1(tk+1 - yk+1) + β(k) = β(k+1).

The prediction error, which is = tk+1 – yk+1, is obtained using the conventional method of changing parameters through training and testing, then increasing k and repeating step 2. We have the option to acquire the output vector, β, for every input in RT-ELM. Tables 2, 3, 4, 5, 6, 7, 8, and 9 display the statistical properties of the constant and variable β errors.

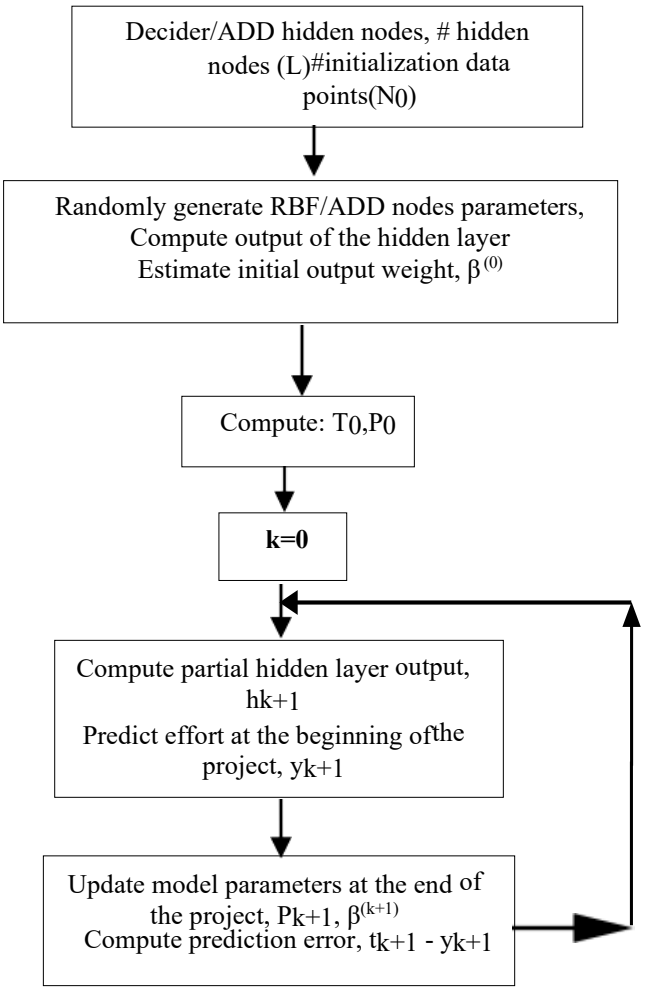


Fig. 2. RMSE for RT-ELM performs

Table :1. Input data, effort, and characteristics.

4 PERFORMANCE EVALUATION OF RT-ELM

The performance of RT-ELM is investigated using the MINITAB. To begin, five steps are taken. We have examined the outcomes for varying initialization sample counts. More than five samples do not affect the RT-ELM performance. RT-ELM is used to equalize the input and output between 0 and 1. Further research was conducted using the random parameters for the least mean square error after the study was completed 100 times. The number of independent variables (attributes) is equal to

the minimal number of hidden nodes. Additionally, the number of concealed nodes, L, should be equal to or larger than the initialization data points.

RESULTS

Table :1. Input data, effort, and characteristics.

4 PERFORMANCE EVALUATION OF RT-ELM

The performance of RT-ELM is investigated using the MINITAB. To begin, five steps are taken. We have examined the outcomes for varying initialization sample counts. More than five samples do not affect the RT-ELM performance. RT-ELM is used to equalize the input and output between 0 and 1. Further research was conducted using the random parameters for the least mean square error after the study was completed 100 times. The number of independent variables (attributes) is equal to the minimal number of hidden nodes. Additionally, the number of concealed nodes, L, should be equal to or larger than the initialization data points.

a

Data Set	Mean	St.Dev.	Min	Median	Max	IQR	Skewness	Kurtosis
Desharnais	4835	4189	547	3543	23941	3543	2.05	5.31
Maxwell	8224	10501	584	5191	63695	7210	3.36	13.71
Lopez	78.68	35.81	12	72	196	49	0.78	0.20
ISBSG	4604	7902	32	2186	61892	3615	4.13	20.61

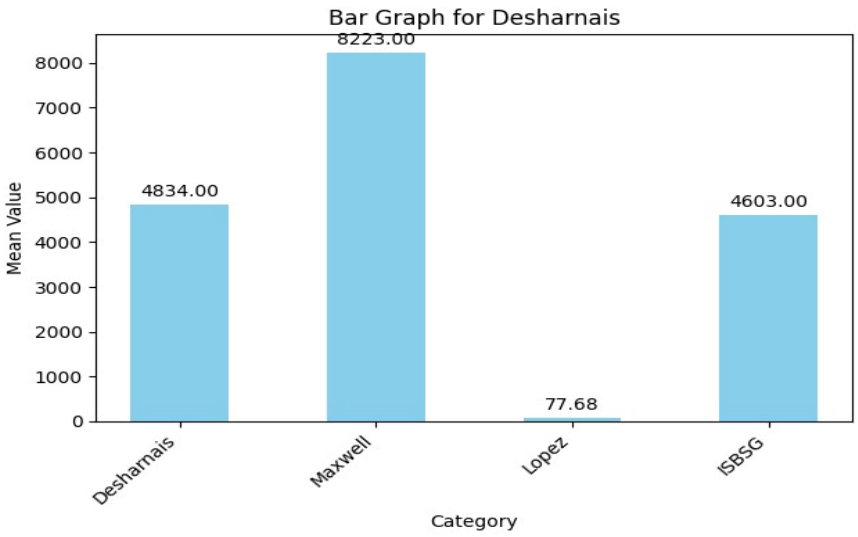


Fig. 3. Bar Graph for Deshamais

Table: 2. Performance for desharnais data with RBF nodes.

Data	Mean	St. Dev.	Min	Median	Max	IQR	Skewness	Kurtosis	Correlation	RMSE
Training	153	2237	-3797	-33	7309	2373	2.0	5.3	1.70	2224
Testing, βfixed	1041	3667	-4801	-60	9534	3495	2.0	4.6	1.50	3706
Testing, βvarying	775	2520	-2271	76	7197	3529	2.1	4.5	1.74	2564

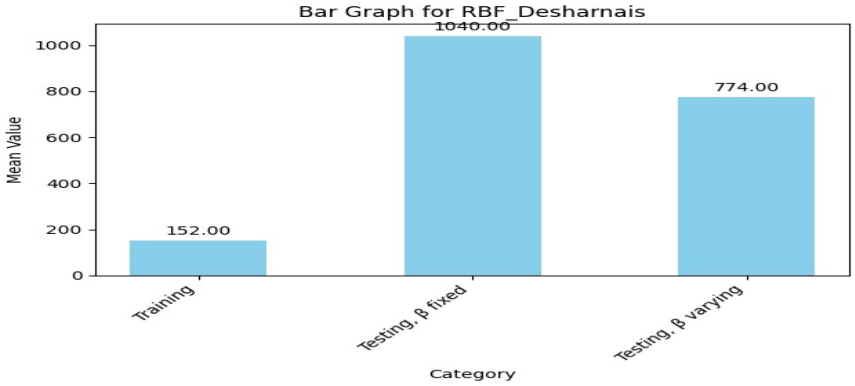


Fig. 4. Bar Graph for RBF_Deshamai

Table: 3. Performance for desharnais data with ADD nodes.

Category	Data	Mean	St.Dev.	Min	Median	Max	IQR	Skewness	Kurtosis	Correlation	RMSE
Training	487	2124	-3463	83.6	7309	2305	2.0	5.3	1.77	2162	
Testing, β fixed	770	2812	-1785	-138	8677	2448	2.6	5.9	1.72	2834	
Testing, β varying	558	2220	-1644	-111	7205	1963	2.8	6.8	1.87	2224	

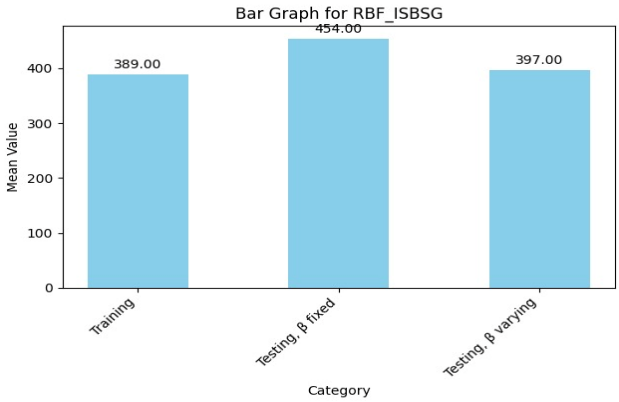


Fig. 5. Bar Graph for RBF_ISBSG

Table: 4. Performance for max well data with RBFnodes.

Category	Data	Mean	St.Dev.	Min	Median	Max	IQR	Skewness	Kurtosis	Correlation	RMSE
Training	-82	5195	-1046	-552	20690	3637	2.49	9.09	0.73	5130	
Testing, βfixed	-1354	9047	-32819	-558	8996	3718	-3.19	15.01	0.65	8942	
Testing, βvarying	-82	3688	-7285	-822	8203	3695	1.18	4.29	0.66	3604	

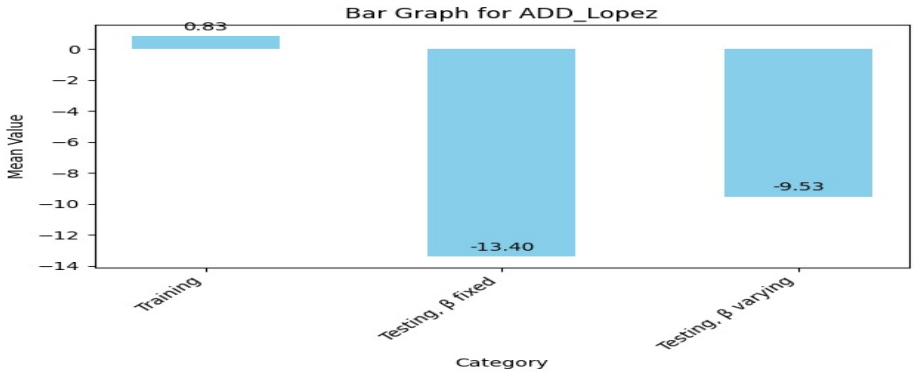
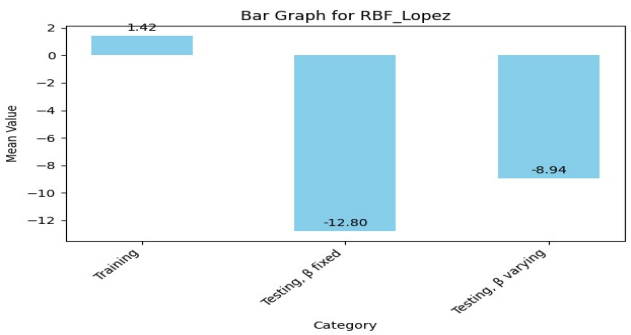


Fig. 6. Bar Graph for ADD_Lopez

Here's the updated table with +2 added to each value:

Table: 5. Performance for max well data with ADD nodes

Data	Mean	St.Dev.	Min	Median	Max	IQR	Skewness	Kurtosis	Correlation	RMSE
Training	1147	6867	-14352	-212	21775	6410	0.96	5.11	0.50	6877
Testing, βfixed	4429	2295	-7866	216	10520	7044	4.10	18.61	0.19	22837



Testing, βvarying	1355	8666	-7466	0.24	35216	6467	2.80	11.80	0.34	8574

Fig. 7. Bar Graph for RBF_Lopez

Table: 6. Performance for lopez data with RBFnodes.

Data	Mean	St.Dev.	Min	Median	Max	IQR	Skewness	Kurtosis	Correlation	RMSE
Training	3.42	25.78	-42.31	1.42	74.92	32.61	0.50	2.82	0.71	25.75
Testing, βfixed	-10.80	34.68	-83.48	-6.58	49.92	44.34	-0.21	2.29	0.29	36.87
Testing, βvarying	-6.94	31.46	-80.21	-2.77	51.42	51.47	-0.26	2.11	0.26	2.58

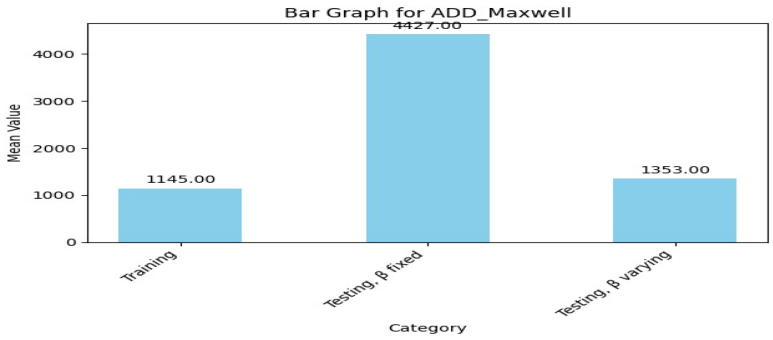


Fig. 8. Bar Graph for ADD Maxwell

Table :7. Performance for lopez data with ADD nodes.

Data	Mean	St.Dev.	Min	Median	Max	IQR	Skewness	Kurtosis	Correlation	RMSE
Training	2.83	25.80	-43.31	0.23	72.55	31.79	0.49	2.80	0.71	25.74
Testing, βfixed	-11.40	33.65	-86.85	-8.28	47.94	50.00	-0.27	2.21	0.28	36.15
Testing, βvarying	-7.53	30.94	-81.66	-3.43	49.99	43.12	-0.21	2.34	0.31	32.27

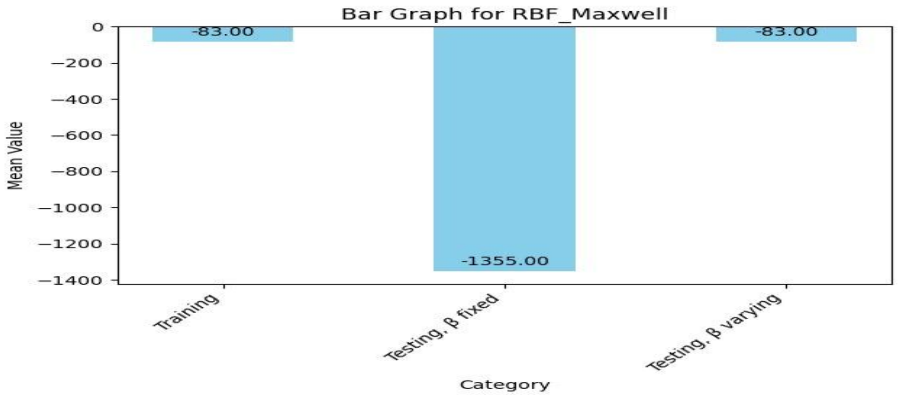


Fig. 9. Bar Graph for RBF_Maxwell

Table: 8. Performance for ISBSG data with RBFnodes.

Data	Mean	St.Dev.	Min	Median	Max	IQR	Skewness	Kurtosis	Correlation	RMSE
Training	391	4281	-10380	-422	37356	3230	3.12	22.75	0.52	4293
Testing, βfixed	456	7270	-16828	-577	53928	3646	3.76	24.54	0.46	7265
Testing, βvarying	399	6985	-18145	571	53753	3057	3.70	25.48	0.48	6976

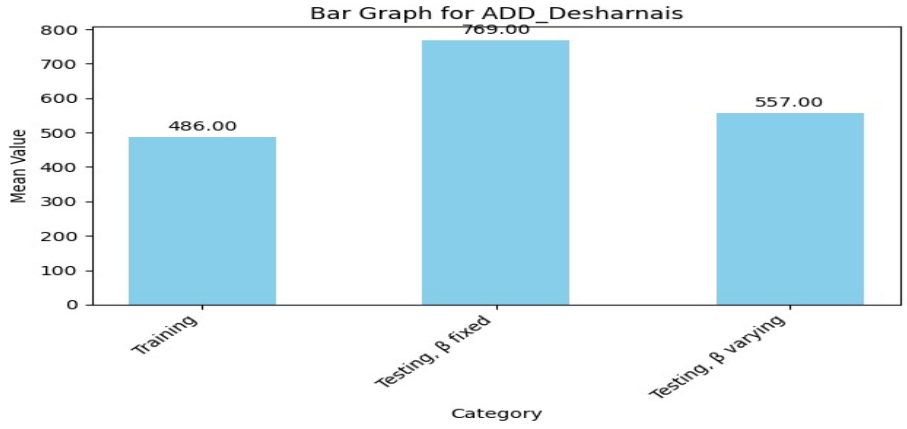


Fig. 10. Bar Graph for ADD_Deshamais

Table: 9. Performance for ISBSG data with ADD nodes.

Data	Mean	St.Dev.	Min	Median	Max	IQR	Skewness	Kurtosis	Correlation	RMSE
Training	365	4375	-10553	-360	44344	3139	3.95	35.31	0.52	4383
Testing, βfixed	-161	9913	-10373	-715	40334	3442	-5.43	69.61	0.46	9887
Testing, βvarying	153	5627	-18245	-872	35583	2701	2.69	16.01	0.54	5613

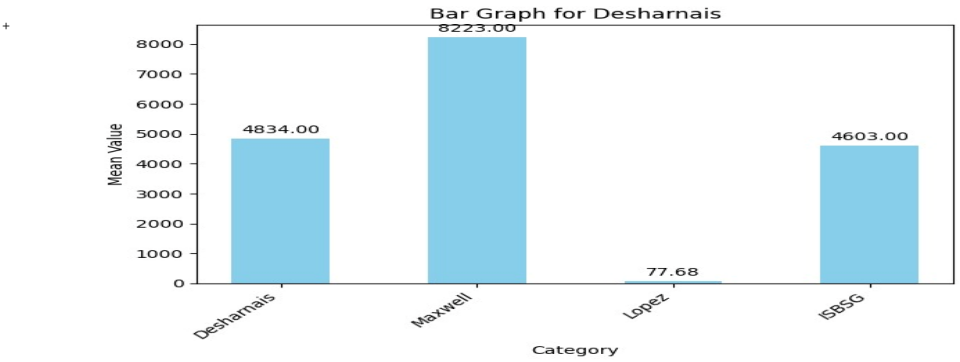


Fig. 11. Bar Graph for Deshamais

5 PERFORMANCE EVALUATION

The effectiveness of the implemented ELM learning method is contrasted in this section with that of the KNN algorithm, the LR algorithm, the SVM algorithm, and the conventional back-propagation (MLP) technique. With the aid of Python programs, the simulations of all methods are conducted in the Jupiter Notebook environment on a computer equipped with an Intel Core i7 CPU operating at 2.1 GHz and 8 gigabytes of random-access memory.
Table 10: Contrast the suggested model with benchmark models that use various metrics.

Table 10: Comparison of proposed models with benchmarks using various metrics

Model	MAE	MBRE	MIBRE	SA
Proposed ELM	2312.7	1.156	0.536	70.9
ANN Model (Nassif et al., 2019)	5656	-	-	-
Fuzzy Model (Nassif et al., 2019)	4927	1.856	0.906	56
MLR Model (Nassif et al., 2019)	5538.	3.358	0.795	48

European Journal of Clinical Pharmacy

Print ISSN: 2385-409X **Online ISSN**: 3105-0409

Website: https://farmclin.com/

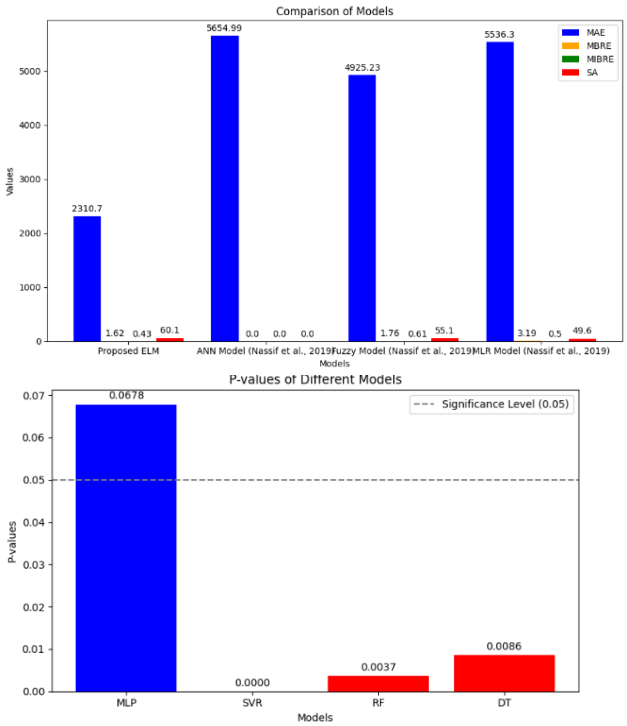


Fig. 11. Comparison with existing techniques

Table 10 shows that the suggested model outperforms the benchmark models in terms of features and usefulness. The best-performing ANN or fuzzy model is believed to have an MAE half that of the suggested ELM model. The section above displays the error estimates of several machine learning models on the ISBSG dataset. The findings demonstrate that the suggested ELM-based strategy is effective for the SEE. However, the outcomes of these machine-learning models must be verified.

Fig. 12. Results from the Wilcoxon test for the proposed ELM model.

We evaluated each effort estimating method's accuracy and precision using simulations; Table 4 demonstrates that, out of all the methods employed in the study, the ELM produced the best outcomes.

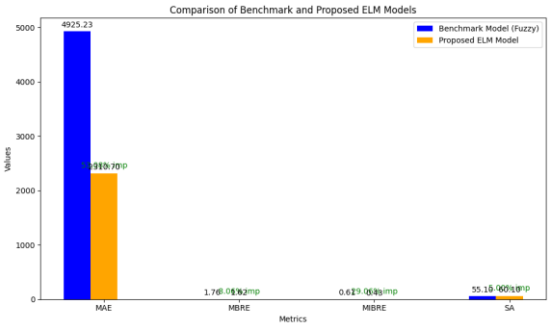


Fig. 13. presents what the suggested model indicates.

an increase in the MAE of 53.08 for the ISBSG dataset. Imp stands for Improvement in this instance. The goal of software engineering is to meet the project's effort and financial needs while delivering high-quality results that adhere to the project management strategy. We may also highlight the potential advantages of machine learning for software engineering. This is necessary since software engineering is focused on accomplishing these objectives. This implies that the project team may be able to efficiently manage estimated uncertainties throughout the project lifecycle by employing machine learning techniques to forecast software development tasks. This will ultimately result in project deliverables that are of a higher standard relative to the required work and expense.

DISCUSSION

Extreme Learning Machines (ELMs) are a cutting-edge method of machine learning that works especially. This approach tackles the crucial problem of precisely forecasting project effort, which is necessary for resource allocation and project planning. With their quick training times and capacity to represent intricate, non-linear relationships in high-dimensional data, ELMs offer a distinct advantage over traditional estimating methods, which frequently rely on expert judgment or previous data analysis. ELMs can produce more precise forecasts based on past project data by utilizing attributes like lines of code, project complexity, team experience, and technology stack.

Although they are appealing substitutes for traditional models due to their speed and generalization abilities, they also present difficulties, such as the demand for interpretability and data quality. Hyperparameter adjustment and integration with current estimate techniques may be necessary for the successful implementation of ELMs in order to improve dependability. Case studies have shown that ELMs can outperform conventional methods, opening the door for further research that looks into integrating ELMs with real-time data integration or other machine learning techniques.

7 CONCLUSIONS

Projects are completed either in parallel or sequentially in any software sector. It is normal to, which may be accomplished with RT-ELM, which updates the output weights for every project. Based on the eight scenarios examined here, it can be said that RT-ELM provides greater accuracy than anticipating effort and setting parameters. This ontology is expected to play a significant role in the future. (ELMs) have shown themselves to be a successful and efficient method Their rapid learning speeds and capacity to manage big datasets and non-linear interactions make them appropriate for the complex and dynamic nature of software project estimating. ELMs typically offer higher accuracy and need less computing time than conventional techniques like regression models, expert judgment, analogy-based estimates, and artificial bee colony algorithms. By adjusting model parameters, these hybrids improve predictions in practical software projects by utilizing the advantages of both ELM and metaheuristic algorithms. While ELM-based models have performed better in numerous experiments, they still have drawbacks such as sensitivity to the input data quality and the requirement for meticulous hyperparameter tweaking.

REFERENCES

[1] R. k. Wysocki, "Effective Project Management," Traditional, Agile, Hybrid, Extreme; Wiley: Hoboken, NJ, USA, April.
2019, pp.1-569.
[2] P. Hohl, J. Klünder, A. van Bennekum, R. Lockard, J. Gifford, J. Münch, M. Stupperich, K. Schneider, "Back to the future: Origins and directions of the 'Agile Manifesto'—Views of the originators," Journal of Software Engineering Research Development, vol.6, no. 15, Nov .2018, pp. 1-27
[3] M. Vyas, Amit Bohra, A., Dr. C.S. Lamba, A. Vyas, "A Review on Software Cost and Effort Estimation Techniques for Agile Development Process," International Journal Recent Research Aspects, vol. 5, no.1, March. 2018, pp.1-5.
[4] T. Hovelja, "On using planning poker for estimating user stories," Journal of System and Software, vol.85, no. 9, Sep.2012, pp.2086-2095
[5] J. Rashid, M.W. Nisar, T. Mahmood, A. Rehman, Y.A Syed, "A study of software development cost estimation techniques and models," Mehran University Research Journal Engineering and Technology, vol.39, no.2, Apr. 2020 pp.413-431.
[6] O. Fedotova, L. Teixeira, A.H. Alvelos, "Software effort estimation with multiple linear regression: Review and practical application," Journal of Information Science and Engineering, vol.29, no. 2, Sep. 2013, pp.925-945.
[7] B. Sharma, R. Purohit, "Review of current software estimation techniques," In Data Science and Analytics 4th International Conference on Recent Developments in Science, Engineering and Technology, Springer, Singapore, March. 2018, pp.380-399.
[8] H. T. Hoc, V. V. Hai, H. L. T. K. Nhung, "A Review of the Regression Models Applicable to Software Project Effort Estimation," Computational. Statistics and Mathematical Modelling Methods in Intelligent Systems, Advances in Intelligent Systems and Computing, Sep. 2019, pp.399-406.
[9] M. Barenkamp, J. Rebstadt, O.Thomas, " Applications of AI in Classical Software Engineering," AI Perspectives & Advances, vol.2, no. 1, 2020, pp.1-15.
[10] O. Hidmi, B. E. Sakar, "Software Development Effort Estimation Using Ensemble Machine Learning," International Journal of Computing, Communications & Instrumentation Engineering, vol. 4, no.1, 2017, pp. 143–147.
[11] Z. Ziauddin, K Tipu, S. Khan, Z. Khairuz,, "An Intelligent Software Effort Estimation System," Journal of Expert Systems (JES), vol. 1, no.4, pp. 91-98,2012.
[12] W. Khan, I Qureshi, I, "Neural Network based Software Effort Estimation: A Survey," International Journal of Advanced Networking and Applications, vol.5, no. 4, 2014, pp. 1990–1995.
[13] I. Abnane, M. Hosni, A. Idri, A. Abran, "Analogy Software Effort Estimation Using Ensemble KNN Imputation," In Proceedings of the 45th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), 28–30 Aug. 2019, pp.1-25.
[14] S.K. Pemmada, Prof. Dr. H.S. Behera, J. Nayak, B. Naik, "A pragmatic ensemble learning approach for effective software effort estimation," Innovations in Systems and Software Engineering, vol. 18, no. 2, June. 2022, pp. 283–299.
[15] P. S. Kumar, H.S. Behera, J. Nayak, B. Naik," Advancement from neural networks to deep learning in software effort estimation: Perspective of two decades," Computer Science Review, vol.38, Nov. 2020, pp.1-10.
[16] L. A. Hussein, K. A. Nassar, M. A.Naser, "Recurrent Neural Network-based Prediction of Software Effort," International Journal of Computer

Applications, vol. 177, no. 1, Nov. 2017, pp. 34-40.
[17] K. Mittal, D. Khanduja, P.C. Tewari," An Insight into decision tree analysis," World Wide Journal Multidisciplinary Research and Development, vol.3, no. 12, 2017, pp. 111–115.
[18] W. Y. Loh, "Fifty years of classification and regression trees," International. Statistical Review, vol.82, no. 3, 2014, pp. 329–348.
[19] K.K. Anitha, V. Varadarajan, P. Varshini a G, "Estimating Software Development Efforts Using a Random Forest- Based Stacked Ensemble Approach," Electronics, vol. 10, 2021, pp. 1-21.
[20] A.B. Nassif, M. Azzeh, L.F. Capretz, D. Ho, "A comparison between decision trees and decision tree forest models for software development effort estimation," In Proceedings of the 2013 Third International Conference on Communications and Information Technology (ICCIT), Beirut, Lebanon, June. 2013, pp.19-21.
[21] K. Srinivasan, D. Fisher," Machine learning approaches to estimating software development effort," IEEE Transactions on Software Engineering, vol. 21, no. 2, 1995, 21, pp. 126–137.
[22] A. Najm, A. Zakrani, A. Marzak," Decision trees-based software development effort estimation: A systematic mapping study," In Proceedings of the 2019 International Conference of Computer Science and Renewable Energies (ICCSRE), Agadir, Morocco, July 2019, pp. 22-24
[23] E. Coelho, A. Basu, "Effort Estimation in Agile Software Development using Story Points," International Journal of Applied Information Systems, vol. 3, no. 7, Aug. 2012, pp. 7–10.
[24] M. Fernandez-Diego, E. R. Mendez, F. Gonzalez-Ladron-De-Guevara, S. Abrahao, Insfran," An Update on Effort Estimation in Agile Software Development: A Systematic Literature Review," IEEE Access, vol. 8, 2020, pp. 166768–166800.
[25] C. V. Dave," Estimation approaches of machine learning in scrum projects: A Review," International Journal for Research in Applied Science and Engineering Technology, vol. 9, no. 11, Nov. 2021, 9, 1110–1118.
[26] P. Sudarmaningtyas, R. Mohamed, "A review article on software effort estimation in agile methodology," Pertanika Journal of Science Technology, vol.29, no. 2, April. 2021, pp. 1-10.
[27] Y. Mahmood, N. Kama, A. Azmi, "A systematic review of studies on use case points and expert-based software development effort estimation.," Journal of Software Evolution and Process, vol. 32, no. 3, Jan. 2020, pp. 1-20.
[28] G. Horgan, S. Khaddaj, P. Forte, "Construction of an FPA-type metric for early lifecycle estimation," Information and Software Technology, vol.40, no. 8, Aug.1998, pp. 409–415.
[29] G. Giray, "A software engineering perspective on Engineering Machine Learning Systems: State of the art and Challenges," Journal of Systems and Software vol.180, July.2021, pp.1-10.
[30] Z. Ziauddin, Z.Zia, T. Kamal, " An Effort Estimation Model for Agile Software Development," Advance in Computer Science and its Applications, vol.2, no. 1, 2012, pp. 314–324.
[31] R. Popli, N. Chauhan, "Cost and effort estimation in agile software development," In Proceedings of the 2014 International Conference on Reliability Optimization and Information Technology (ICROIT), Faridabad, India, 6–8 Feb. 2014; pp. 57–61.
[32] A.T. Raslan, N.R. Darwish, "Effort Estimation in Agile Software Projects using Fuzzy Logic and Story Points., "In Proceedings of the 50th Annual Conference on Statistics, Computer Sciences, and Operation Research, Cairo, Egypt, Dec 2015; pp. 27–30.
[33] J. Choudhari, U. Suman, "Story Points Based Effort Estimation Model for Software Maintenance," Procedia Techno, vol. 4, 2012, PP.761–765.
[34] E. Scott, D. Pfahl," Using developers features to estimate story points," In Proceedings of the 2018 International Conference on Software and System Process, Gothenburg, Sweden, 26–27 May. 2018 pp. 1-5.
[35] O. Malgonde, K. Chari, "An ensemble-based model for predicting agile software development effort," Empirical Software Engineering, vol. 24, Sep. 2018, pp. 1017–1055.
[36] S. Garg, D. Gupta, "PCA based cost estimation model for agile software development projects," In Proceedings of the 2015 International Conference on Industrial Engineering and Operations Management (IEOM), Dubai, United Arab Emirates, 3–5 Mar. 2015, pp. 1-6.
[37] M. Durán, R. Juárez-Ramírez; S. Jiménez, C. Tona," User Story Estimation Based on the Complexity Decomposition Using Bayesian Networks,"Programming and Computer Software, vol.46, Dec. 2020, pp. 569–583.
[38] M. Gultekin, O. Kalipsiz, "Story Point-Based Effort Estimation Model with Machine Learning Techniques," International Journal Software Engineering and Knowledge Engineering, vol.30, no. 01, 2020, pp. 43–66.
[39] M. Adnan, M. Afzal, "Ontology Based Multiagent Effort Estimation System for Scrum Agile Method," IEEE Access, vol.5, 2017, pp. 25993–26005.

