Journal of Theoretical and Applied Information Technology
31st December 2024. Vol.102. No. 24

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

NEMAEP: A NOVEL ENSEMBLE MACHINE LEARNING FRAMEWORK FOR ACCURATE EFFORT ESTIMATION IN SOFTWARE PROJECTS

PRATEEK SRIVASTAVA[1, 2], NIDHI SRIVASTAVA[1], RASHI AGARWAL[3] AND PAWAN SINGH[4]

[1]Amity Institute of Information Technology, Amity University Uttar Pradesh, Lucknow Campus, Lucknow, India
[2]Department of Information Technology, School of Engineering and Technology (UIET), CSJM University, Kanpur, India
[3]Department of Computer Science and Engineering, Harcourt Butler Technical University, Kanpur, India
[4]Department of Computer Science and Engineering, Amity University Uttar Pradesh, Lucknow Campus, Lucknow, India

E-mail: [1,2]prateeksri976@gmail.com , [1]nsrivastava2@lko.amity.edu , [3]dr.rashiagrawal@gmail.com, [4]psingh10@lko.amity.edu

ABSTRACT

The increasing complexity of software engineering projects has made accurate effort estimation a formidable challenge. We introduce the NEMAEP framework to address existing methods shortcomings and enhance precision. This framework integrates ensemble learning techniques, specifically the CatBoost gradient boosting algorithm, with Grid Search cross-validation for optimizing hyperparameters. Our robust predictive methodology substantially improves the accuracy and reliability of software effort estimation. We evaluated NEMAEP against well-established regression techniques, including support vector regressor, decision tree, random forest, and multi-layer perceptron. Two datasets were utilized: Cocomo81, comprising 63 projects, and a more extensive China dataset, containing 499 projects. We assessed performance using MAE, RMSE, and R^2. The NEMAEP methodology demonstrated superior predictive capabilities, achieving accuracy rates of 97% for Cocomo81 and 99% for the China dataset. When applied to the China dataset, the model produced MAE and RMSE scores of 0.0154 and 0.0213, respectively. In Cocomo81, the values were 0.013 and 0.0641. This innovative strategy allows for the effective maximization of resource utilization in software project management. This is achieved by providing project managers with a data-driven tool to navigate modern software development projects.

Keywords: *Software Project Management, Software Effort Estimation, Machine Learning, Ensemble Learning, Grid Search*

1. INTRODUCTION

During the software development process, software effort estimation is crucial to the success of the project. In building a software project, estimates of schedules, costs, and manpower play a crucial role. Software effort estimation predicts the effort needed to maintain and develop a software system based on unknown, incomplete, unreliable, and inconsistent input data. A significant amount of attention has been paid to software effort estimates since the mid-1970s. The accuracy of the predictions was essential since inaccurate predictions can lead to overestimation of resources and dissipation of funds, while underestimation can lead to an excess of spending. Software technology has continually extended its significance and size, thus adding to its complexity, and making it more difficult to get an accurate estimate of the effort.

Several techniques are available for estimating software efforts, such as algorithmic approaches, expert opinions, measurement through analogies, and artificial intelligence [1]. Numerous mathematical algorithms are involved, including COCOMO, the slim model, and end-user use cases. Estimating a project with expert judgment involves analyzing and utilizing the expert's experience. A comparison of homogeneous historic projects with current development models is known as estimation by analogy. Software effort estimation techniques using machine learning have been a renowned technology for more than two decades.

By utilizing prior knowledge from prior projects to construct regression models, machine learning generates estimates of software project effort using prior projects. Feature and effort correlation in projects seems difficult to model

using statistical methods or traditional parametric models as software project complexity grows. As software effort estimation has made significant progress, it is expected that techniques will emerge to estimate the effort of drastically modifying code, allowing programming skills and tools to keep pace with advances. Machine learning might be preferred for this scenario over traditional methods since it has the capability of accessing historical data and learning from it, as well as adapting to a wide range of variation that occurs in software development [2]. Due to their accuracy and efficiency, these techniques are efficient at dealing with complicated data.

Sometimes, it is difficult to determine which effort estimators are most effective using machine learning techniques. Since the ranking of estimators may change when they are compared based on modified conditions. When several estimators are combined, the result is superior to a single estimate. As a result of the above claim, ensemble learning is recognized as a method of predicting or estimating performance by combining multiple learning algorithms. The use of an ensemble approach for estimating the amount of effort required for software development achieves a much higher degree of accuracy by combining various algorithms and predicting the best outcome [3].

This paper aims to estimate software effort utilizing ensemble learning techniques and machine learning techniques using Grid Search CV. The following are some of the highlights of the research work:

1. To develop and evaluate a comprehensive machine learning framework for the various datasets, integrating a diverse range of algorithms including traditional methods (SVR, Decision Trees), advanced ensemble techniques (Random Forests, CatBoost), neural networks (MLP), with a focus on enhancing prediction accuracy through Grid Search CV hyperparameter optimization.

2. To implement and assess the effectiveness of a two-step robust data preprocessing methodology, combining the Robust Scaler for feature scaling and the Interquartile Range (IQR) method for outlier detection and removal, in improving the overall performance and reliability of multiple machine learning models applied to the various datasets.

3. To conduct an extensive comparative analysis of various machine learning algorithms, contrasting their performance both with and without hyperparameter optimization, using multiple evaluation metrics (MAE, RMSE and R2 Score) to identify the most effective approach for predictive modelling on the various datasets. This analysis aims to provide insights into the relative strengths of different algorithms, with particular emphasis on the performance of advanced techniques like CatBoost, and to visualize these comparisons for clear interpretation.

The paper consists of the following sections: Section 2 describes the related work on methods for calculating the effort required to complete a software project. Section 3 explains the proposed model along with the framework. Section 4 shows the analysis of the results along with the graphical representation. Section 5 presents a comparison of our findings with those from extant studies. Section 6 provides conclusions.

2. RELATED WORK

Mukesh Mahadev, et al. [4] have explored and implemented genetic programming approaches to estimate software effort. Genetic programming has been studied and adapted for the task of effort estimation in detail and has been reported clearly. Understanding bioinspired evolutionary algorithms is the first step, followed by exploring metrics that can be used to assess their performance. To analyse the prediction model, metrics like Pred25 and MMRE were implemented using genetic programming. To test the predictive model, multiple datasets of various sizes were used. More reliable accuracy is obtained by K-fold validation. Results show good accuracy for the model built using GP.

An ensemble of heterogeneous stacked ANN and SVR is proposed by Somya Goyal [5] to estimate effective effort. The proposed model is then empirically compared with base learners to determine its accuracy. For the comparison, datasets from the PROMISE repository were selected. The accuracy measures used for the comparison were MAR and MMRE. According to the proposed model, MAR and MMRE are reduced by 50.4% and 54.6%, respectively, over base models. According to the experiments, the proposed heterogeneous stacked ensemble is the most statistically advantageous model among the candidates for SEE. Multiple comparisons are validated statistically using the Friedman test.

Stacking regularization analogy-driven methodology (SABE) is proposed by Anupama Kaushik, et al. [6] to improve the accuracy of effort

prediction based on ABE. This technique is based on machine learning, which is the core of SABE. Multi-model stacking provides better estimation accuracy than a single model since it takes advantage of multiple models' capabilities. A comparison is made between the proposed method and the currently available solution functions. To assess the results, MMRE, MdMRE, PRED, and SA are used as the evaluation criteria. SABE showed promising results in comparison to earlier studies on almost all evaluation criteria.

Several machine learning algorithms have been recommended by Mizanur Rahman, et al. [7] including k-nearest neighbour regression, support vector regression, and decision trees. Software development industries now use these methods for estimating software to overcome the limitations of parametric and conventional estimation techniques. To assess the effectiveness of the established procedure, the authors examined a dataset created by Edusoft Consulting LTD. The three commonly used performance evaluation measures, MAE, MSE, and R squared, are the foundation for these experiments. A comparison of decision trees and other techniques shows that decision trees are more accurate at predicting effort.

This research aims to apply the LSTM algorithm and examine how accurate it is at estimating software effort. To decide which model is superior, Farah B. Ahmad, et al. [8] compared the results with previous work. China and Kitchenham data sets were analysed using the LSTM algorithm. An evaluation of the model's accuracy is based on RMSE, MAE, and R-squared. A China dataset has an RMSE of 0.016, MAE of 0.019, and R-squared of 0. 972.Kitchenham dataset has an RMSE of 0.017, MAE of 0.058, and R-squared of 0.896. LSTM algorithm performed better on both data sets than other algorithms, proving their superiority.

This study aims to reduce the gap between actual and predicted software effort for future projects by providing a more accurate software effort estimation model. All datasets cannot be predicted better by a single machine learning algorithm. As a result of using voting estimators, Beesetti Kiran Kumar, et al. [9] proposed the ensembling of regression models to reduce error rates over those provided by a single machine learning algorithm. The ensemble model showed a lower error rate. In the area of machine learning, the average between different predictors had a positive impact on the output, demonstrating the crucial role played by these predictors in optimizing software effort estimation. Results showed that there was an improvement in the performance between ensemble and individual models on different datasets.

3. PROPOSED METHODOLOGY

In this section, the NEMAEP model is discussed in a detailed manner. The designed framework comprises of the following stages as shown in fig 1.

3.1. Installing Libraries: In this step, libraries and modules are imported that are needed for data manipulation, visualization, and implementing various machine learning algorithms. Matplotlib and seaborn are used to visualize data, and scikit-learn implements machine learning models, with pandas and numpy for manipulating data. Additionally, libraries like Cat Boost are imported for their respective gradient-boosting algorithms.

3.2. Data Loading: In this step, the dataset under investigation is loaded from a specified file path into a Pandas Data Frame, a two-dimensional tabular data structure that facilitates efficient data manipulation and analysis. The paper works with two different datasets, draws inspiration from the prior studies. Table 1 lists the data sets including the source repository, number of projects, number of attributes and target variable.

Table 1. Dimensions Of The Datasets

Datasets	Source Repository	No. of Attributes	No. of Projects	Target Variable
China	Promise	19	499	Effort
Cocomo81	Promise	17	63	Effort

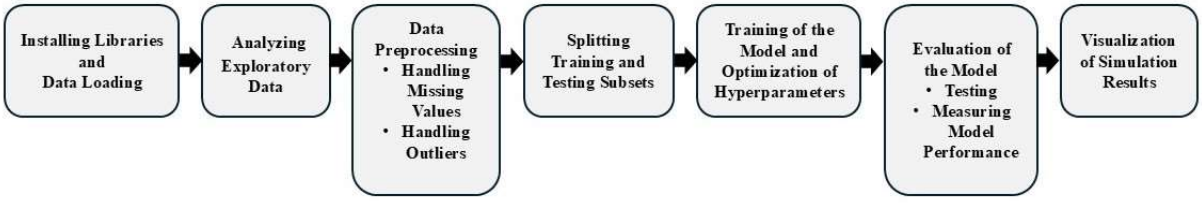

Fig 1. Proposed Architecture Of NEMAEP

3.3. Analyzing Exploratory Data: An important part of data analysis is exploratory data analysis, where the dataset is thoroughly explored to gain insights into its structure, statistical properties, and potential issues. Various EDA techniques are employed, such as checking data types, computing summary statistics (mean, median, standard deviation, etc.), identifying unique values, and detecting missing data points. Preprocessing of the data is guided by this step, which assists in understanding the data.

3.4. Data Preprocessing

3.4.1 Handling Missing Values: If the EDA reveals the presence of values that are missing from the dataset, relevant techniques are employed to deal with these values. Several approaches can be used to replace missing values with estimated values, or to remove missing values (drop rows or columns with missing values).

3.4.2 Handling Outliers: Outliers are samples that vary substantially from the entire dataset and negatively impact machine learning models' performance. A robust scaling technique (Standard Scaler) and the Interquartile Range (IQR) method are employed to identify and handle outliers. Standard Scalers remove the mean and scale the variance to a unit, which reduces the effect of outliers on standardizing features. The IQR method identifies outliers as values lying more than 1.5 times the IQR away from the first or third quartiles, and the corresponding rows are removed from the dataset.

3.5 Splitting Training and Testing Subsets: The pre-processed input is segmented according to training and testing categories to examine how well machine learning models generalize. We have used the 80-20 split ratio for the training and testing set. The paper proposed a variety of experiment considering five machine learning based regression algorithms namely, SVR, DT, RF, MLP and CatBoost on both the datasets. We have used the advance ensemble machine learning algorithm known as CatBoost for accurate effort estimation and it provided outstanding results in comparison to other algorithms. The model is taught with a training set, and its efficiency is assessed using a testing set.

3.5.1 CatBoost Algorithm: It is a gradient boosting algorithm that is capable of handling categorical features and handles them during the training phase rather than during the preprocessing period [10]. By permuting categorical variables, a new technique called Ordered Boosting is implemented, which produces a numerical interpretation. In this method, the category information is preserved while the gradient-boosting technique is used to boost the model's performance. The regularization technique employed by CatBoost prevents overfitting. To prevent the model from becoming too complex during the training process and fitting the training data too closely, these techniques introduce penalties or constraints. A regularized model is more robust to unseen data and helps to generalize it. The algorithm minimizes the loss function iteratively using gradient descent to construct an ensemble of trees. A loss function is calculated at each iteration, and a tree is fitted to the negative gradient according to current predictions. The learning rate determines step sizes during gradient descent. The process is repeated as long as a convergent threshold is met, or a predetermined quantity of trees has been added. The predictions of each tree within an ensemble are merged by CatBoost when it makes predictions. A highly accurate and reliable model is thus derived by aggregating predictions.

3.6 Training of the Model and Optimization of Hyperparameters: Training of regression models and optimization of hyperparameters are performed at this step. The Grid Search CV technique is used to tune hyperparameters for each model. A hyperparameter is a model-based setting that cannot be learned from data but has a significant impact on the model's performance. Every pair of hyperparameter values is cross validated to determine model accuracy. According to a particular scoring metric (e.g., negative mean absolute error), the hyperparameter combination that yields the best performance is selected. The machine learning techniques with the chosen parameter values are shown in table 2.

Table 2. Parameter Settings For Various Machine Learning Approaches

ML Techniques	Parameter Values
SVR	Penalty parameter (C): 0.1 to 10 Kernel coefficient (gamma): 0.01 to 1 Kernel type: linear and radial basis function
DT	Maximum tree depth: unrestricted, 5 levels, and 10 levels Minimum samples for split: 2, 5, and 10 Minimum samples per leaf: 1, 2, and 4
RF	Number of trees: 100, 200, and 500 Maximum tree depth: unrestricted, 5

	levels, and 10 levels Minimum samples for split: 2, 5, and 10 Minimum samples per leaf: 1, 2, and 4
MLP	Hidden layer configurations: (50), (100), and (50, 50) neurons Activation function: ReLU and hyperbolic tangent Learning rate schedule: constant and adaptive
Cat Boost	Number of iterations: 100, 200, and 500 Tree depth: 3, 5, and 10 levels Learning rate: 0.01, 0.1, and 0.2

3.6.1 Hyperparameter Tuning using Grid Search CV: Machine learning models are constructed by specifying hyperparameters or variables. Various combinations of parameter values are evaluated against a specified evaluation metric, such as grid search, to discover the best hyperparameters [11]. A hyperparameter is useful because it permits the individual to customize it to the system's precise requirements. In addition to Grid Search, Grid Search CV employs a cross-validation process. The most widely used validity assessment technique is K-fold cross-validation. In this iterative process, the train data is successively partitioned into k segments. Training is performed using k-1 partitions, while one partition is kept for testing each iteration. During the next iteration, the next partition will be regarded as test data and k-1 as train data, etc. The model will be recorded each time it is run, and at the end, the average performance will be given.

3.6.2 Working of Grid Search Enabled CatBoost Algorithm: The steps of Grid Search Enabled CatBoost Algorithm are shown below.

Step 1: Start with the dataset and perform the necessary preprocessing.
Step 2: Set the initial hyperparameters for the CatBoost model.
Step 3: Build the CatBoost ensemble model using the initial hyperparameters.
Step 4: Fit the CatBoost model and evaluate its performance.
Step 5: Update the sample weights based on the model's performance.
Step 6: Use Grid Search CV to search through the defined hyperparameter grid.
Step 7: Find the next set of hyperparameter combinations from the grid.
Step 8: Check if the stopping criteria (e.g., maximum iterations, performance threshold) are met.
Step 9: If not, repeat steps 3-8 with the new hyperparameters.
Step 10: Once the stopping criteria are met, obtain the optimal hyperparameters.
Step 11: Perform the final performance evaluation using the optimized model.

3.7 Evaluation of the Model

3.7.1 Testing: The models are trained and tuned, then tested on the remaining set. Test data is used by each model to make predictions that are examined in comparison to the target values.

3.7.2 Measuring model performance: Several evaluation metrics are used to quantify each model's performance, including MAE, RMSE, and R-squared. Model predictability and good fit can be evaluated from various perspectives based on these metrics.

For our investigation, to calculate and examine the efficiency of the several regression algorithms, we applied three performance measurements that included mean absolute error (MAE), root mean squared error (RMSE), and coefficient of determination (R2).

3.7.2.1 Mean Absolute Error: The MAE measures the average deviation between the actual and estimated values for all the observations considered. Models with lower MAE values are more accurate. This is computed as shown below:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|y_i - \hat{y}_i\right| \quad (1)$$

3.7.2.2 Root Mean Squared Error: To calculate the MSE, we take the mean square differences based on the true and estimated scores of various sample sets. An evaluation of the RMSE is obtained by computing the square root of the MSE as well as by finding the mean deviation of the residuals. To achieve better performance, a model should have a lower RMSE value. The following formula is used to calculate it:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2} \quad (2)$$

3.7.2.3 Coefficient of Determination: The determination coefficient (R2) determines how close predictions are to the real values. R2 ranges from 0 to 1. When the value obtained is 1, the model fits exactly to the dataset, and when it is negative, the model does not fit well. A model should have a higher R2 value closer to 1 for superior performance. The following formula is used to calculate it:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2}{\sum_{i=1}^{n}\left(y_i - \bar{y}_i\right)^2} \quad (3)$$

In equations (1)-(3), y_i corresponds to actual values, $\hat{y}_i$ corresponds to predicted values, and n represents the overall project.

The values obtained from the above evaluation metrics on both datasets are described in the results and discussion section of the paper.

3.8 Visualization of Simulation Results: Bar plots are used to visualize the model's performance to facilitate comparisons and interpretations. A clear and concise comparison of the relative strengths and weaknesses of each model can be made based on the visualizations that show the evaluated metrics for each model as shown in fig 2(a) and fig 2(b).

4. RESULTS AND DISCUSSION

To address the concerns outlined in section 1, this segment will perform two types of investigations to evaluate the reliability of the suggested framework. The first set will be conducted without adjusting hyperparameters (using default settings), while the second set will involve hyperparameter optimization using grid search cross-validation.

4.1 Model with Default Parameter Tuning: This phase involved comparing various machine learning algorithms (SVR, DT, RF, MLP, and Cat Boost) using default parameter settings without hyperparameter optimization. The aim was to evaluate which algorithm might perform best across different problems without fine-tuning. The machine learning techniques have been applied using default configurations on the training data. More accurate results are obtained from the ML technique that displays the minimum MAE and RMSE values. For R2 values, higher numbers indicate better accuracy. Table 3 shows the optimal performance metrics generated by every technique on each data set using randomly set parameters within the specified range without tuning. Individual models perform differently based on the set of data used as shown in fig 2(a).

4.1.1 Mean Absolute Error (Default Parameter Tuning): The first assessment of machine learning algorithms using Mean Absolute Error (MAE) and default settings showed varying performance levels across the two datasets. In the case of the China dataset the Cat Boost method performed well with an MAE of 0.0157 closely followed by Random Forest (RF) and Decision Trees (DT) with MAEs of 0.0186 and 0.0189 respectively. This indicates that the Cat Boost algorithms ensemble technique effectively captures the underlying patterns, in the China dataset. For the Cocomo81 dataset Random Forest outperformed other algorithms with an MAE of 0.0462 while Cat Boost came in second with 0.0486.

4.1.2 Root Mean Squared Error (Default Parameter Tuning): RMSE analysis using default hyperparameters provided insights into the accuracy and consistency of the algorithms predictions. In the China dataset DT showed the RMSE of 0.0308 closely trailed by RF at 0.0323 and Cat Boost at 0.0375. While Cat Boost had the MAE its higher RMSE indicates some variability in predictions compared to DT and RF. In the Cocomo81 dataset RF outperformed other algorithms with an RMSE of 0.0626 closely followed by Cat Boost with an RMSE of 0.0641.

4.1.3 Coefficient of Determination (Default Parameter Tuning): The R squared value (R2) is a metric to gauge how well each model explains the data. When using default settings all algorithms performed impressively on the China dataset with Decision Trees (DT) achieving the R2 score of 0.9962 closely followed by Random Forest (RF) at 0.9958 and the suggested Cat Boost method, at 0.9944. This suggests that all models, including Cat Boost can explain a portion of the variability in the target variable for this dataset. On the Cocomo 81 dataset RF achieved the R2 score of 0.8025 and the Cat Boost method also performed well at 0.7925. The high R2 scores across both datasets indicate that Cat Boost algorithm offers high predictive power in software effort estimation.

Table 3. Performance Metrics For Various Regression Models Using Default Parameter Tuning

Evaluation Metrics	Dataset	Machine Learning Algorithm				
		SVR	DT	RF	MLP	Proposed Method
MAE	China	0.0694	0.0189	0.0186	0.0614	0.0157
	Cocomo81	0.0951	0.0751	0.0462	0.0973	0.0486
RMSE	China	0.0865	0.0308	0.0323	0.0924	0.0375
	Cocomo81	0.0960	0.1086	0.0626	0.1204	0.0641
R2	China	0.9704	0.9962	0.9958	0.9662	0.9944
	Cocomo81	0.5357	0.4052	0.8025	0.7280	0.7925

4.2 Model with Hyperparameter Tuning: Subsequently, the model employs hyperparameter optimization utilizing the grid search cross-validation technique. During the training phase, the machine learning algorithm is fine-tuned with hyperparameter adjustment on the training dataset. A five-fold cross-validation approach is applied, in which the complete dataset is split into five identical portions. The model undergoes training and testing five times, with every iteration utilizing a different fold for testing and four folds for training. The model's effectiveness is assessed on the test set in each round. The final performance metric is calculated by averaging these five evaluations. Model performance varies depending on the dataset used as shown in fig 2(b). Table 4 presents the optimal performance values for each model and dataset following hyperparameter optimization.

4.2.1 Mean Absolute Error (After Hyperparameter Tuning): Significant improvements in algorithm performance were observed across datasets due to hyperparameter tuning. In the dataset from China the Decision Tree (DT) algorithm showed progress by achieving the Mean Absolute Error (MAE) of 0.0119 after tuning surpassing the previous leader. The Cat Boost method also maintained performance with an improvement to an MAE of 0.0154 showcasing its resilience to parameter adjustments. In the Cocomo81 dataset there was a shift in algorithm rankings as the Cat Boost method exhibited enhancement and achieved the lowest MAE of 0.0130. This improvement highlights the potential of optimizing and adapting the Cat Boost algorithm to data sets effectively. These findings underscore how crucial hyperparameter tuning is for maximizing algorithm performance and suggest that fine tuning enhances the effectiveness of algorithms like Cat Boost, on both datasets.

4.2.2 Root Mean Squared Error (After Hyperparameter Tuning): Hyperparameter optimization resulted in improvements in RMSE for all algorithms and datasets. In the China dataset the Cat Boost technique stood out as the performer with an RMSE of 0.0213 after tuning marking an enhancement from its initial performance. This indicates that tuning greatly boosted the precision and consistency of Cat Boosts predictions on this dataset. RF and DT also saw progress achieving RMSE values of 0.0309 and 0.0345 respectively. In the Cocomo81 dataset RF maintained its lead with a RMSE of 0.0394 while the RMSE value of Cat Boost algorithm remains unchanged, with 0.0641. Despite good performance at 0.0641 for the Cat Boost method on this dataset it suggests that its default settings were already well optimized here.

4.2.3 Coefficient of Determination (After Hyperparameter Tuning): Hyperparameter optimization resulted in excellent R2 scores, for most machine learning algorithms across datasets. In the China dataset the Cat Boost approach showed progress achieving a R2 score of 0.9982 post tuning indicating a significant improvement over its initial performance. This exceptional result suggests that the Cat Boost algorithm effectively explains all variability in the target variable after tuning surpassing traditional methods. In the Cocomo81 dataset Cat Boost also exhibited a boost achieving the R2 score of 0.9758. This notable advancement showcases Cat Boosts capability to optimize and capture patterns within datasets. The varying outcomes across datasets stress the importance of evaluation and tailored optimization, for accurate software effort estimation tasks.

Table 4. Performance Metrics For Various Regression Models After Hyperparameter Tuning

Evaluation Metrics	Dataset	Machine Learning Algorithm				
		SVR	DT	RF	MLP	Proposed Method
MAE	China	0.0694	0.0119	0.0184	0.0487	0.0154
	Cocomo81	0.0951	0.0466	0.0235	0.0497	0.0130
RMSE	China	0.0594	0.0345	0.0309	0.0625	0.0213
	Cocomo81	0.0837	0.0788	0.0394	0.0799	0.0641
R2	China	0.9860	0.9952	0.9962	0.9845	0.9982
	Cocomo81	0.6468	0.6871	0.9214	0.6779	0.9758

The presented NEMAEP methodology offers significant benefits in software effort estimation due to its innovative integration of the CatBoost ensemble learning approach and Grid Search Cross-Validation. This study's greatest achievements lie in its extensive strategy for data preprocessing, particularly robust scaling and outlier detection, and comprehensive parameters-based optimization technique. Despite these achievements, the study has certain limitations. This investigation focuses on only two datasets, so it may reduce the relevance of the findings to a wider group of software development applications. Furthermore, even though the CatBoost algorithm delivered outstanding results, the performance variability among various datasets indicates that the model's efficiency appears to be contextual. Future work could extend the dataset diversity, examine alternative ensemble procedures, and determine the model's feasibility with additional distinct software development domains to further verify and improve the present analysis.

5. COMPARISON WITH EXTANT STUDIES

Our suggested CatBoost ensemble machine learning method delivers enhanced efficiency for various performance metrics relative to earlier investigations in software effort estimation, as shown in Table 5. The presented methodology generates extremely low MAE of 0.0154 and 0.013 for the China and Cocomo81 datasets. This is much smaller than conventional techniques such as LSTM, Stacked Ensemble, and multiple regression techniques. The precision of the method is further demonstrated by the RMSE values of 0.0213 for the China dataset and 0.0641 for the Cocomo81 dataset, respectively. Moreover, our method exhibits exceptional R2 scores of 0.9982 and 0.9758 for the China and Cocomo81 datasets, indicating an extremely high level of model reliability and explanatory power. When contrasted with previous works that used techniques such as Multi-layer Perceptron, Social Network Search algorithms, Particle Swarm Optimization, and various ensemble methods, our approach stands out by providing more accurate and consistent software development effort estimations, thereby offering a more robust solution for project planning and resource allocation in software development contexts..

6. CONCLUSION AND FUTURE SCOPE

The NEMAEP methodology presented in this study greatly improves software effort estimation. This study illustrates how ensemble learning procedures and advanced hyperparameter optimization can significantly improve software project estimation performance. The purpose of this study is to provide empirical evidence that the proposed methodology can enhance prediction capabilities over conventional regression approaches. One of the major advantages of this method is that it integrates modern machine learning techniques into a consistent method for estimating software effort reliably. Even though this research recognizes the limitations of dataset complexity, it lays the groundwork for future exploration into more universal predictions. A fundamental advancement in software engineering is made possible by the NEMAEP framework, which utilizes machine learning to maximize accuracy and reliability in resource allocation.

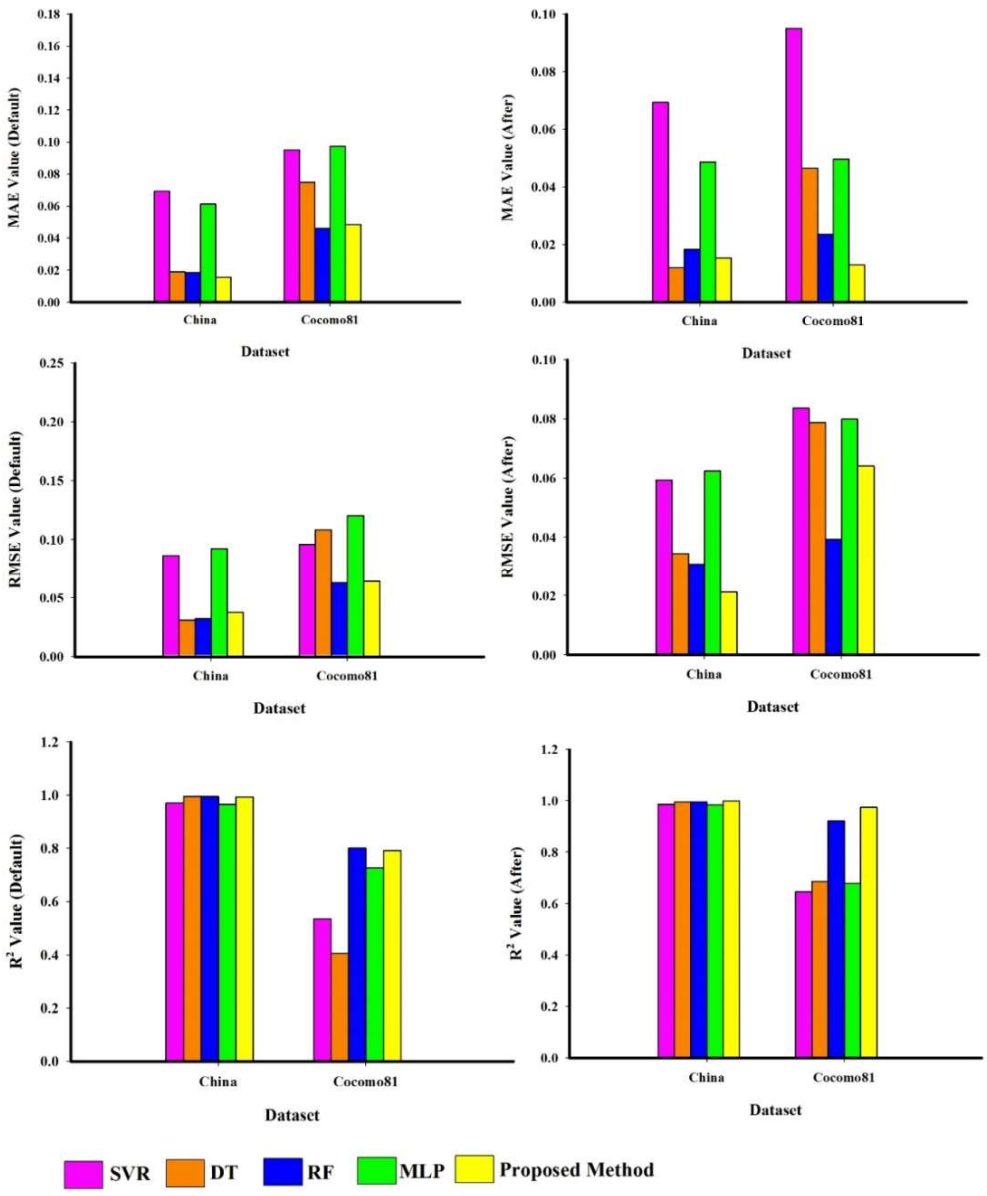

Fig 2(a). Comparison Chart for performance metrics

Fig 2(b). Comparison Chart for performance metrics

using default parameter tuning in various algorithms after hyperparameter tuning in various algorithms

Table 5. Comparative Analysis Of Proposed Work With Previous Work

		MAE		RMSE		R2	
References	**Model/Datasets**	**China**	**Cocomo 81**	**China**	**Cocomo81**	**China**	**Cocomo 81**
[12]	ANFIS (Adaptive Neurofuzy Inference System) and SNS algorithm (Social Network Search)					0.9716	0.5086

[13]	Multi-layer Perceptron Assisted Honey Bidirectional Gated Recurrent Feed Forward Network (Multi-Hbig)	0.0753	0.0763				
[14]	Long Short Term Memory Neural Network (LSTM)					0.951	0.897
[15]	LR+PSO Model (Linear Regression+ Particle Swarm Optimization)		0.128		0.208		0.544
[16]	Ensemble Method (Bagging, Boosting, Voting)						0.9578
[17]	MS-DES (Multi Step Dynamic Ensemble Selection)					0.9845	
[18]	Gradient Boosting						0.88
[19]	Stacked XG Boost	0.3059	0.3262	0.31262	0.33283	-5.9491	-3.404
[20]	LSTM+GS (Long Short Term Memory+ Grid Search)					0.894	
[21]	SVM, RF,DT,SGB,NB,MLP,LR,KNN	0.0204		0.0679		0.7453	0.3644
[22]	Stacked Ensemble		0.0227808				
[9]	Ensembling of Regressor Models using Voting Estimator		0.1466		0.2297		
[8]	LSTM					0.972	
[23]	Stacked LSTM		0.087		0.2	0.981	0.189
[24]	RF,SVM,DT,Neurelnet,Ridge,LASSO, Elasticnet,Deepnet		0.0538		0.0944	0.9736	0.8212
[25]	RF,CART,KNN,MLP,SVR,AdaBoost	0.0243	0.0557			0.9473	0.8582
[26]	Gradient Boosting Regressor					0.93	
[27]	ANN (Artificial Neural Network)						0.946
[28]	Deepnet, Neuralnet, SVM, RF			0.0443		0.8548	
[29]	ANFIS (Adaptive Neurofuzy Inference System)		1.32		1.14		
	Proposed Method	**0.0154**	**0.013**	**0.0213**	**0.0641**	**0.9982**	**0.9758**

REFERENCES

[1] Srivastava P, Srivastava N, Agarwal R, Singh P. An Intelligent Framework for Estimating Software Development Projects using Machine Learning. IJRITCC. 2023 May;11(5):160-9.

[2] Tan AJJ, Chong CY, Aleti A. REARRANGE: Effort estimation approach for software clustering-based remodularisation. Inf Softw Technol. 2024; 176:1-19.

[3] Kumar KH, Srinivas K. An improved analogy-rule based software effort estimation using HTRR-RNN in software project management. Expert Syst Appl. 2024; 251:124107.

[4] Mahadev KM, Gowrishankar G. Estimation of Effort in Software Projects using Genetic Programming. Int J Eng Res Technol. 2020;9(7).

[5] Goyal S. Effective Software Effort Estimation using Heterogenous Stacked Ensemble. In: 2022 IEEE International Conference on Signal Processing, Informatics, Communication and Energy Systems (SPICES); 2022; Thiruvananthapuram, India. p. 584-588.

[6] Kaushik A, Kaur P, Choudhary N, Priyanka. Stacking regularization in analogy-based software effort estimation. Soft Computing. 2022 Feb 1;26(3):1197–216.

[7] Rahman M, Roy PP, Ali M, Gonc T, Sarwar H. Software effort estimation using machine

learning technique. Int J Adv Comput Sci Appl. 2023;14(4).
[8] Ahmad FB, Ibrahim LM. Software Development Effort Estimation Techniques Using Long Short Term Memory. In: 2022 International Conference on Computer Science and Software Engineering (CSASE); 2022 Mar 15; IEEE; 2022. p. 182-187.
[9] Beesetti KK, Bilgaiyan S, Mishra BS. Software Effort Estimation through Ensembling of Base Models in Machine Learning using a Voting Estimator. Int J Adv Comput Sci Appl. 2023;14(2).
[10] Ibrahim AA, Ridwan RL, Muhammed MM, Abdulaziz RO, Saheed GA. Comparison of the CatBoost classifier with other machine learning methods. Int J Adv Comput Sci Appl. 2020;11(11).
[11] Zakrani A, Hain M, Idri A. Improving software development effort estimating using support vector regression and feature selection. IAES Int J Artif Intell. 2019 Dec 1;8(4):399.
[12] Manchala P, Bisi M. TSoptEE: two-stage optimization technique for software development effort estimation. Cluster Computing. 2024 Apr 12:1-20.
[13] Anitha CH, Parveen N. Deep artificial neural network based multilayer gated recurrent model for effective prediction of software development effort. Multimed Tools Appl. 2024 Jan 25:1-27.
[14] Iordan AE. An Optimized LSTM Neural Network for Accurate Estimation of Software Development Effort. Mathematics. 2024 Jan 8;12(2):200.
[15] Jayadi P, Ahmad KA, Cahyo RZ, Aldida JD. Particle Swarm Optimization-based Linear Regression for Software Effort Estimation. J Inf Syst Technol Eng. 2024 Jun 19;2(2):261-8.
[16] Oshaibi MF, AlKhanafseh M, Surakhi O. Software Effort Estimation using Ensemble Learning [Preprint]. 2024.
[17] Jadhav A, Shandilya SK, Izonin I, Muzyka R. Multi-Step Dynamic Ensemble Selection to Estimate Software Effort. Appl Artif Intell. 2024;38(1):2351718.
[18] Meharunnisa, Saqlain M, Abid M, Awais M, Stević Ž. Analysis of software effort estimation by machine learning techniques. Ingénierie Syst Inf. 2023;28(6):1445-1457.
[19] Varshini P, Kumari KA. Software Effort Estimation using Base Ensembled Regression Techniques and Principal Components Regression as Super Learner [Preprint]. 2023 Mar 16 :1-24.
[20] Marco R, Ahmad SS, Ahmad S. An Improving Long Short Term Memory-Grid Search Based Deep Learning Neural Network for Software Effort Estimation. Int J Intell Eng Syst. 2023 Jul 1;16(4):164-180.
[21] Jadhav A, Shandilya SK. Reliable machine learning models for estimating effective software development efforts: A comparative analysis. J Eng Res. 2023;11(4):362-376.
[22] Rao K, Pydi B, Naidu P, Prasann U, Anjaneyulu P. Ensemble Learning Approach for Effective Software Development Effort Estimation with Future Ranking. Adv Distrib Comput Artif Intell J. 2023; 12:1-16.
[23] Ahmad FB, Ibrahim LM. Software effort estimation Based on long short term memory and stacked long short term memory. In: 8th International Conference on Contemporary Information Technology and Mathematics (ICCITM); 2022; Mosul, Iraq. p. 165-170.
[24] Alhamdany F, Ibrahim L. Software Development Effort Estimation Techniques: A Survey. J Educ Sci. 2022;31(1):80-92.
[25] Marco R, Syed Ahmad SS, Ahmad S. Bayesian Hyperparameter Optimization and Ensemble Learning for Machine Learning Models on Software Effort Estimation. Int J Adv Comput Sci Appl. 2022;13(3):419-429.
[26] Kumar PS, Behera HS, Nayak J, et al. A pragmatic ensemble learning approach for effective software effort estimation. Innov Syst Softw Eng. 2022; 18:283-299.
[27] Mohsin ZR. Application of artificial neural networks in prediction of software development effort. TURCOMAT. 2021 Oct 5;12(14):4186-202.
[28] Varshini, Priya, Kumari K A, Janani D, Soundariya S. Comparative analysis of Machine learning and Deep learning algorithms for Software Effort Estimation. J Phys Conf Ser. 2021; 1767:1-11.
[29] Varshini P, Kumari KA, Janani D, Soundariya S. Comparative analysis for estimating development effort of software projects using MODA, ANFIS and COCOMO. J Emerg Technol Innov Res. 2018;5(6):667-674.