Vol. 06, No. 02, pp. 161 –168 (2025)
ISSN: 2708-0757

JOURNAL OF APPLIED SCIENCE AND TECHNOLOGY TRENDS

www.jastt.org

Software Effort Estimation Using Stacking Ensemble and Bayesian Optimization

Muhammad Yusuf, Daniel Siahaan*

Department Informatics Engineering, Institut Teknologi Sepuluh Nopember,Surabaya Indonesia, 6025231066@student.its.ac.id, daniel@its.ac.id

Corresponding: daniel@its.ac.id

Abstract

Accurately estimating software costs is a vital step in ensuring the successful completion of a software project. There is a need for estimation techniques that ensure projects are completed on time, within budget, and with the desired quality. Accurate estimation plays a crucial role in crafting realistic budget plans and ensuring that projects are completed on time with sufficient resources. When estimations are precise, teams can spot potential issues early, distribute resources more effectively, and handle risks with greater confidence. This research focuses on boosting the reliability of software effort estimation by applying a stacking method enhanced with Bayesian hyperparameter optimization. It leverages three core algorithms SVM, Random Forest, and Decision Tree each fine-tuned using the proposed approach. Evaluations across 11 public datasets reveal noteworthy improvements, ranging from 0.2 to 0.5. A significance test confirms the model's strong performance, showing a p-value greater than 0.5, which indicates that the results are statistically meaningful. These findings suggest that combining stacking with Bayesian tuning holds promise for refining software effort predictions. It can serve as a valuable reference for future project planning across diverse modelling approaches.

Keywords: *Software Estimation, Ensemble Learning, Stacking, Hyperparameter Tuning, Bayesian Optimization, Machine Learning, Model Performance.*

Received: June 14th, 2025 / Revised: July 25th, 2025/ Accepted: August 02nd, 2025 / Online: August 07th, 2025

I. INTRODUCTION

Project managers must estimate software costs accurately to allocate resources, define meaningful targets, and identify risks that could impact software development [1] [2]. Precise cost prediction allows project managers to forecast quantifiable requirements, such as staff, hardware, and apps, and estimate the time needed to complete a software development project [3] [4] More precise budget management and appropriate decision-making can help the estimating attempt succeed in line with the aims achieved [5] [6].

Efforts to ensure software development accuracy must consider several factors and criteria. These efforts are often expressed in terms of person-hours or person-months [7] [8]. Inaccurate estimates can cause problems, such as delays, inadequate resource allocation, and failure [9] [10]. Accurate estimation is crucial for the success of a software project, ensuring timely completion, staying within budget, and meeting project objectives. Additionally, accurate estimation helps avoid project failure, which can result in cost and resource constraints

Estimates in software development have traditionally relied on techniques such as expert judgment, parametric estimation, and top-down or bottom-up methodologies [11] [12]. These approaches are commonly used in the industry, particularly by teams managing complex and dynamic projects, but they have drawbacks, especially in situations with many unknowns [13] [14]. As a result, more adaptive and accurate approaches, such as machine learning and ensemble methods, are gaining increasing attention [15] [16] [17].

Several previous studies, such as method in [18] have used machine learning to address software effort estimation and shown better results than traditional effort estimation methods. Focused on four regression models: AdaBoost, Gradient Boosting, Linear Support Vector, and Random Forest Regression. The results showed that Random Forest Regression provided the best performance with a high accuracy of 0.80. On the other hand, Meharunnisa et al. [19] specifically compared linear regression and random forest in software effort estimation. Their findings showed that Random Forest achieved the highest accuracy, reaching 0.90. A similar result was

doi: 10.38094/jastt62310

reported in [20] who observed a significant improvement in estimation accuracy using machine learning techniques, with Random Forest showing excellent results.

The research does not stop there; machine learning techniques, including ensemble learning, continue to evolve. The ensemble learning approach can be used to improve predictive performance. Ensemble learning is a group decision-making system that combines predictions from multiple models to generate a better prediction [21] [22] [23]. Research in [24] applied a boosting ensemble for software effort estimation by combining neural network models and decision trees. The results showed that the ensemble model provided significant performance improvements compared to single models. Similarly, [25] explored two ensemble models combined with feature selection, finding that the stacking model delivered the best results, with an ROC area of 0.973.

Based on the studies explained, although some research shows improved accuracy with machine learning methods, there is still limited research combining stacking ensemble learning with Bayesian hyperparameter tuning to improve software effort estimation accuracy. This study is motivated to utilize the stacking technique, which combines the results from several base models to improve prediction accuracy by leveraging the strengths of these models. Meanwhile, Bayesian hyperparameter tuning allows for automatic adjustment of model parameters to achieve more optimal results, accounting for uncertainty in the training process. Therefore, applying stacking ensembles is expected to provide the best accuracy performance [26] At the same time, Bayesian tuning can offer better computational optimization, making the resulting model for software effort estimation superior to previous models [27].

The primary contributions are outlined as follows:

- A novel approach to enhance the accuracy of software effort estimation by integrating Bayesian hyperparameter optimization with stacking ensemble learning. This method optimizes the configurations of several basic models, leveraging their predictive strengths to achieve better performance.
- This study demonstrates that improving the accuracy of software development effort estimation yields superior project management outcomes compared to previous research. Furthermore, it highlights the potential to elevate client satisfaction by planning software development more efficiently and reliably.

The study is structured into several key sections. Section I presents the background, motivation, and research objectives. In Section II, the pertinent literature is reviewed, the theoretical underpinnings are established, and prior research is examined. The research methodology, including the experimental design, evaluation criteria, and procedures used during the study, is described in Section III. The experiment's findings are examined in Section IV, emphasizing the advancements made possible by ensemble learning and hyperparameter optimization. Section V concludes with a summary of the findings and recommendations for possible future study directions.

II. Literatur Review

Research conducted by Sakhrawi et al. [18] discusses the problem of estimating software development efforts that are not accurate in making change requests, thus causing project planning failures and cost estimation problems. This research applies four regression methods namely AdaBoost Regressor, Gradient Boosting Regressor, Linear Support Vector Regression, and Random Forest Regression. The results showed that random forest regression has good results with MAE 0.040, MSE 0.045, and RMSE 0.215.

In [28] the authors discuss the issue of inaccurate software development effort estimation, which leads to project failures and financial losses due to poor cost estimation. This research proposed to use Adaptive Neuro-Fuzzy Inference System (ANFIS) applied with COCOMO dataset. The results show that this technique is superior to the traditional methods previously used and produces more accurate estimates.

In another relevant study, [24] highlights the inaccuracies in software effort estimation that often lead to project failure. This research highlights that traditional methods such as Case-Based Reasoning (CBR) require expert intuition, which makes them less effective for handling large and complex data sets. So this research optimizes CBR method combined with Genetic Algorithm to select the best parameters for estimation. the results found that this method is more accurate and efficient compared to traditional CBR.

In [29] the authors address the issue of low estimation accuracy in software development efforts, which leads to project delays, cost overruns, and inefficient resource allocation. To determine the best model for development effort estimation, they contrasted eight machine learning techniques. According to the study, Random Forest has continuously shown the most reliable and superior performance in terms of estimation accuracy.

Lastly, [30] focused on the inaccuracies of software cost estimation when using single models or basic meta-learners. This study applied a Stacking Ensemble model combined with Grid Search for parameter configuration. The results showed that the Stacking Ensemble model with Grid Search provided more accurate cost estimation than single models.

Based on previous research, ensemble models with a stacking approach have performed better than single models in software cost and effort estimation. Therefore, this study examines how well the Stacking Ensemble model, combined with Bayesian Hyperparameter Tuning, can improve estimation accuracy compared to traditional estimation methods that rely on single models. Given that this approach has proven to yield more optimal results in previous studies, we expect that this combination will address the limitations of single models and provide more accurate estimation results. We will further elaborate on the procedures used in this study in the following sections.

III. METHOD

This work's approach is divided into three primary stages: planning, carrying out, and assessing. The stages are evaluation, algorithm implementation, and data collection and preparation. The experimental design of this work is depicted in Figure 1.

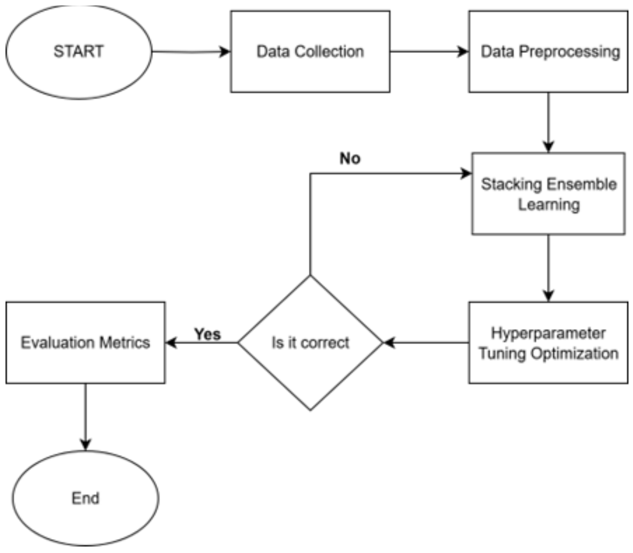

Fig 1. Methodology

A. Data Collection and Prepocessing

Eleven distinct datasets from earlier studies are used in this investigation. The Albrecht, COCOMO81, Desharnais, and Kemerer datasets are often used for software work estimation. With 499 and 407 items, respectively, this study's Chinese and Finnish datasets are relatively big. With a taxonomy of the datasets and the independent and target variables for each dataset included in Table I, this section provides a thorough overview of the datasets. Every dataset has characteristics that show the elements influencing the software project's overall development effort. This is significant since the research emphasizes the variety of the datasets and their differing attributes, including the quantity of files collected by various organizations and the data collected in the form of Lines of Code (LOCs) and Function Points (FPs). Exploratory data analysis (EDA) was used to examine the datasets' structure, content, and quality. This included determining the number of unique, null, and missing values. [31] [32] [33].

TABLE I. DESCRIPTION DATASET

Dataset	No of Attrubute	No of Observasi	Target X	Target Y
Albertch	8	24	FPAdj, RawFPCOunt, AJFP,Input,Output, Inquire and File	Effort
China	11	499	Interface, Included, Modified, Removed,Resource,Dev, Type,Duration, PDR_APP, PDR_UFP, NPDR_APP,NPDU_UFP	Effort
Cocomo81	15	63	Acap,Aexp,Pcap,Vecp,Mcdp,Tool,Sced,Loc,Rely,Data,Cplx,Time,Stor,Virt,Turn	Effort
Deshernaise	9	81	TeamExp,ManagerExp,Length,Languange,Transaction,Entities,Adjusment,PointAdjust,PointNonAdjust.	Effort
Finnish	10	407	Size, pgpspp90, StafCost,T01-T021,Int, top,Int_prp,Int_app,Int_out,Out_	Workup
Kemeker	6	15	Hardware, Duration, AdjFP, RAWFP, KSLOC, and Language	Effort
Kitchenham	6	145	Project, Actual Duration, Client.code, Modified. First, function.points estimates	Actual Effort
Maxwell	26	62	Time, Size, Duration, Har, Dia, Ifc, Source, Nlan, Total Lines, T01-T15	MM
Miyazaki	7	48	FORM, FILE, ESCRN, EFORM, EFILE, KLOC, and SCRN	Measured Effort
Telecom	4	18	ACT_DEV, ACT_TEST, CHNOS, FILES	Act
Nasa93	3	18	KLOC, Methodology(MB)_	Act_Effort

B. Stacking Ensemble Learning

Ensemble learning is a technique that uses many models to improve prediction accuracy. Preparing the dataset for training and testing begins with preprocessing, which includes operations like feature extraction, missing data treatment, and normalization. The dataset is split into training and testing sets at an 80:20 ratio. During the training stage, three distinct modelsRandom Forest (RF), Support Vector Machine (SVM), and Decision Tree (DT) are created. Each model is trained using data to produce predictions. Once the models are trained, they generate predictions on the testing data. All three models' results are combined for a more stable and accurate prediction. Depending on the specific ensemble technique used, this combination can be done through methods like averaging or voting. The combined forecasts are then sent to a meta-learner, a higher-level model that maximizes the merging of the underlying models' predictions.

To improve performance, hyperparameter tuning is performed, where parameters of the models, such as the number of trees in Random Forest or the kernel in SVM, are adjusted to find the best configuration. The model's performance is assessed after tuning to identify the best outcomes. If not, more changes are performed. Lastly, as seen in Fig. 2, the model's performance is evaluated using measures such as accuracy, MEA, MAE, or RMSE to ascertain how well the ensemble model performs in comparison to single model

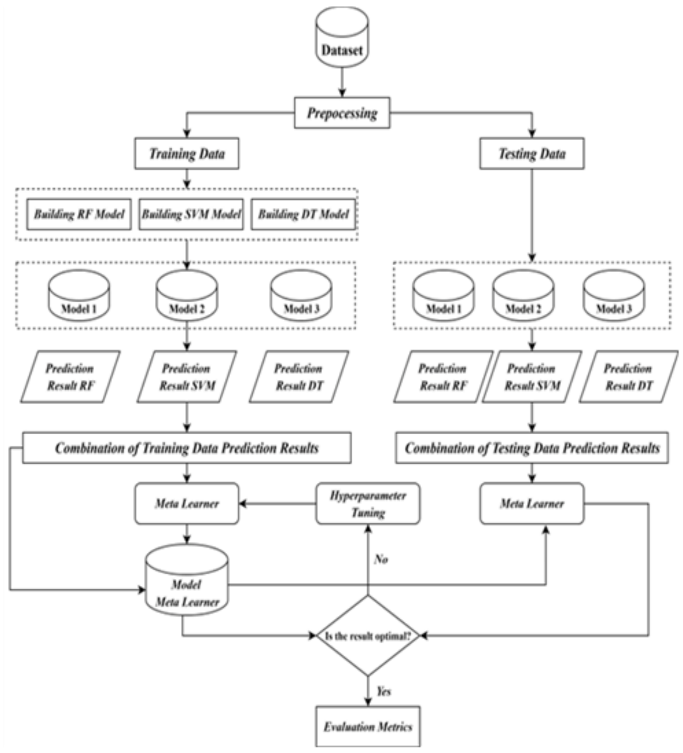

Fig 2. Proposed Stacked Ensemble Learning

C. Bayesian Hyperparameter Tuning

Once the stacking ensemble model has been constructed, its performance is optimized through hyperparameter adjustment. .In this study, Optuna is used for hyperparameter tuning as it can efficiently identify the best parameters, saving time compared to other methods. The parameters optimized in this study include those listed in Table II.

TABLE II. PARAMETERS VALUE BAYESIAN TUNING

ALGORITMS	PARAMETER
SVR	Penalty parameter (C): 0.01 to 1000 Kernel coefficient (gamma): 0.0001 to 0.1 Kernel type: 'rbf'
RF	Number of estimators: 50 to 200 Maximum tree depth: 2 to 10 Minimum samples for split: 2 to 10
DT	Maximum tree depth: unrestricted, 5 levels, 10 levels Minimum samples for split: 2, 5, 10 Minimum samples per leaf: 1, 2, 4
STACKING	Base models: SVR, Random Forest, Decision Tree (DT) Meta-model: Best base model (selected based on Optuna)

This step seeks to determine the optimal parameters for each model in the ensemble (Random Forest, SVM, and Decision Tree) to increase accuracy and stability. In this study, Optuna is used for hyperparameter tuning as it can efficiently identify the best parameters, saving time compared to other methods Each model has specific parameters, such as tree depth for Decision Trees or kernel type for SVM, which are adjusted for better performance. The meta learner settings are also customized to improve the integration of the predictions.

- **Step 1** : Define the Search Space (For each base modeland for the meta-learner)
- **Step 2** : Create an Objective Function (Instantiate and cross-validate the model on the training set 5-fold CV)
- **Step 3** : Run the Optuna Study
- **Step 4** : Retrive Best Hyperpameters (after optimatizon, extract *study.best_params* for each models)
- **Step 5** : Refit Base Models (Train each base learner on the full training set using its Optuna-optimized hyperparameters)

D. Evaluation Metrics

To evaluate the performance of the built model, several measures are used to test its prediction accuracy. These metrics show how effectively the model captures the relationship between the input data and the final output.

- R-squared metric is used to determine how well a model fits your data. It's a statistical measure of how well a regression model. This approximates the actual data in the context of regression.

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

- Mean Absolute Error (MAE) is the average of the total absolute error at each data point. The absolute error itself is the absolute difference between the actual target value and the value predicted by the model.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \quad (2)$$

- Mean Squared Error (MSE) is the average sum of the squares of the differences. The differences are between the actual target value and the predicted value.

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

- Root Mean Squared Error (RMSE) is the standard deviation of the predicted deviation.

$$RMSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \quad (4)$$

IV. RESULT AND DISCUSSION

This section further discusses our research on software effort estimation through the proposed purposed method of stacking ensemble learning optimized by Bayesian hyperparameter tuning. The experimental results begin with a discussion of evaluating the performance of the model with a single model without the proposed method, then evaluating the performance of the model with the proposed method. Finally, the overall results achieved from this research are discussed.

A. Model Without Proposed Method

In this study, we compared three of the best machine learning models selected based on previous research, namely Support Vector Machine (SVM), Random Forest (RF), and Decision Tree (DT). These three prototypes were regarded as the most efficient and sturdy among the five prototypes examined earlier. The selection of these models is crucial in supporting the stacking ensemble process. Combining the strengths of these three models makes the stacking ensemble process more powerful because these models have proven to perform optimally on various datasets. Compared to approaches that use more models, the results of this study show that using these three models as the foundation for the ensemble gives better performance. Without hyperparameter tuning and using default settings, we can evaluate the baseline performance of each model without further tuning**.**

From Table III, generally, RF performs better on most datasets, with higher R^2 values compared to SVM and DT on many datasets. However, SVM performs better on the Cocomo 81 and China datasets, with higher R2 values than RF and DT. On the other hand, DT shows weaker performance on datasets such as Cocomo 81, Deserhanis, and Kitchenham, indicating weaknesses in this model on specific datasets.

In Table IV, which shows the MAE values, RF consistently has lower MAE values than SVM and DT on most datasets, tindicating that RF is more accurate in predicting values with fewer errors. Although SVM performs well on the China and Cocomo 81 datasets, RF remains superior in many cases, particularly on the Deserhanis and Maxwell datasets, showing RF's advantage in prediction precision.

TABLE III. RESULT R^2 WITHOUT PROPOSED METHOD

DATASET	SVM	RF	DT
Albertch	0.9047	0.8211	0.8057
China	0.6437	0.7002	0.6502
Cocomo81	-3.0020	0.2949	-0.0035
Deshernaise	0.5464	0.4165	0.5406
Finnish	0.3690	0.4790	-0.4757
Kemeker	0.3786	0.2683	0.0966
Kitchenham	-1.1230	0.8640	0.7770
Maxwell	0.7081	0.8379	0.4311
Miyazaki	0.5289	0.7310	0.2849
Telecom	0.0350	0.0369	0.2006
Nasa93	-0.3870	0.1789	0.1755
Albertch	0.9047	0.8211	0.8057

Regarding MSE, RF performs better on most datasets, with lower MSE values than SVM and DT, especially on the Cocomo 81 and Maxwell datasets. However, DT shows less stable performance on other datasets, particularly on the Telecom and NASA 18 datasets, indicating weaknesses in handling more complex data.

In Table V, which shows the RMSE values, RF performs best on most datasets again. While SVM and DT show lower RMSE values on specific datasets like China and Cocomo 81, RF overall has more consistent and lower RMSE values on most datasets. This indicates that RF is better.

B. Model With Proposed Method

In this section, we evaluate the performance of the machine learning models with Stacking Ensemble and Bayesian Hyperparameter Tuning applied. The stacking method, with the advantage of a meta learner built from base models, can enhance overall performance. Additionally, by incorporating Bayesian optimization, the parameters of each base model can be fine-tuned to further improve the ensemble's effectiveness.

Table VI shows that the stacking ensemble with Bayesian hyperparameter tuning has improved R^2 for most datasets compared to single models. For example, in the Albert dataset, the stacking ensemble achieved a significantly higher R^2 value of 0.9140 compared to SVM (0.9047), RF (0.8211), and DT (0.8057). This shows that the combination of models has improved prediction accuracy.

TABLE IV. RESULT MAE WITHOUT PROPOSED METHOD

DATASET	SVM	RF	DT
Albertch	0.0721	0.1312	0.1245
China	0.0460	0.0220	0.0350
Cocomo81	0.0460	0.0222	0.0160
Deshernaise	0.0250	0.0027	0.0280
Finnish	0.0435	0.0231	0.0423
Kemeker	0.0335	0.0345	0.0365
Kitchenham	0.0070	0.0250	0.0260
Maxwell	0.0277	0.0345	0.0623
Miyazaki	0.0435	0.0231	0.0423
Telecom	0.0334	0.0245	0.0376
Nasa93	0.0570	0.0676	0.0735

China dataset shows significant improvement, achieving an R^2 score of 0.7725 through the stacking ensemble substantially outperforming each individual model. The ensemble models performed better than the single models. However, in some datasets like Deserhanise and Telecom, the improvements in R^2 are more modest, indicating that the stacking ensemble's impact varies across different datasets. Nevertheless, the ensemble model has consistently outperformed the single models, confirming the effectiveness of stacking and hyperparameter tuning.

In Table VII, we compare the results of the same stacking ensemble approach with results from previous research [29]. The previous research shows different performances for the single models and the stacking ensemble when applied to various datasets. Meanwhile, in Table VIII, we compare the MAE values with state of the art results, where this study demonstrates superior performance compared to previous research.

TABLE V. RESULT MSE WITHOUT PROPOSED METHOD

DATASET	SVM	RF	DT
Albertch	0.0230	0.2130	0.1230
China	0.0060	0.0048	0.0063
Cocomo81	0.0360	0.0030	0.0041
Deshernaise	0.0170	0.0187	0.0175
Finnish	0.0325	0.0245	0.0543
Kemeker	0.0425	0.0725	0.0643
Kitchenham	0.0222	0.0170	0.0320
Maxwell	0.2456	0.0324	0.0434
Miyazaki	0.0325	0.0245	0.0543
Telecom	0.0045	0.0789	0.0234
Nasa93	0.0425	0.0243	0.0256

TABLE VI. RESULT RMSE WITHOUT PROPOSED METHOD

DATASET	SVM	RF	DT
Albertch	0.0623	0.2345	0.1678
China	0.0070	0.0690	0.1670
Cocomo81	0.0600	0.0560	0.0246
Deshernaise	0.1675	0.1435	0.1656
Finnish	0.0879	0.0689	0.0790
Kemeker	0.0679	0.0489	0.0690
Kitchenham	0.1339	0.1434	0.1345
Maxwell	0.0413	0.0576	0.0789
Miyazaki	0.0879	0.0689	0.0979
Telecom	0.0689	0.0800	0.0916
Nasa93	0.0776	0.0456	0.0967

For instance, in the Albert dataset, previous research reported an R^2 of 0.8852 for SVM, 0.9202 for RF, and 0.1877 for DT, while the stacking ensemble achieved 0.9140. Although the stacking ensemble result is similar to the previous research, it is noteworthy that the RF model performed slightly better in the previous research, with an R^2 of 0.9202, compared to the 0.8211 achieved in the current study. This suggests that while stacking still benefits, the specific choice of base models can influence the results.

For Cocomo 81, previous research had a much lower performance in R^2 with SVM and RF (both negative values), with the stacking ensemble still outperforming the single models at 0.4944. This highlights that stacking can significantly improve the performance even when the single models perform poorly on specific datasets.

TABLE VII. RESULT R^2 WITH PROPOSED METHOD

DATASET	SVM	RF	DT	PROPOSED METHOD
Albertch	0.9047	0.8211	0.8057	**0.9140**
China	0.6437	0.7002	0.6502	**0.7725**
Cocomo81	-3.0200	0.2949	-0.0325	**0.4944**
Deshernaise	0.5464	0.4165	0.5406	**0.5505**
Finnish	0.3690	0.4790	-0.4757	**0.7410**
Kemeker	0.3786	0.2683	0.0966	**0.4000**
Kitchenham	-1.1230	0.8640	0.7770	**0.8506**
Maxwell	0.7081	0.8379	0.4311	**0.7785**
Miyazaki	0.5289	0.7310	0.2849	**0.8413**
Telecom	0.0350	0.0369	0.2006	**0.7877**
Nasa93	-0.3870	0.1789	0.1755	**0.4083**

TABLE VIII. COMPARISON MAE STATE OF THE ART

DATASET	CHINA	COCOMO81	MAXWELL	KEMEKER
RF [29]	0,0184	0,0235	0,0260	0,0487
DT [29]	0,0366	0,0168	0,0547	0,0801
SVM [29]	0,0441	0,0478	0,0501	0,0688
SENSE [32]	0,0437	-	0,0405	0,0215
NEMAEP [31]	0,0154	0,0130	-	-
FCNN [33]	0,0310	0,1158	0,1157	0,2021
GWO+FCNN [33]	0,0218	0,0130	0,0037	0,0003
PROPOSED METHOD	**0,0058**	**0,0062**	**0,0038**	**0,0042**

Another interesting observation is the Miyazaki dataset, where the stacking ensemble results from both studies are the same (0.8413). However, previous research had a very low R^2 for SVM (-7.5835) and a modest result for RF (0.2435). This suggests that stacking can stabilize the performance even in cases where single models might produce unreliable or erratic predictions.

C. Intertability Analysis

The performance of single models is contrasted with the suggested approach for effort estimate in this interpretability investigation. By comparing the absolute residuals of each model pair, the Wilcoxon statistical test is used to assess how well these models compare to one another. The test results are illustrated in Table IX.

TABLE IX. PARAMETERS VALUE BAYESIAN TUNING

MODEL COMPARISION	**P VALUE**	**STATISTICAL SIGNIFICANCE**
RF Vs PROPOSED METHOD	0,0432	SIGNIFICANT ($P < 0.05$)
DT Vs PROPOSED METHOD	0,0132	SIGNIFICANT ($P < 0.05$)
SVM Vs PROPOSED METHOD	0,0354	SIGNIFICANT ($P < 0.05$)

When compared to single models, the suggested approach performs well and is statistically significant, according to the data above. It is important to note that a p-value greater than 0.5 does not indicate statistical significance. However, as shown in the results from Table IX, the p-value is less than 0.5, which confirms the statistical significance of the suggested approach. This finding supports the idea that the proposed approach is a better substitute for effort estimation across different datasets.

D. Discussion

The results presented in this study show that Random Forest (RF) outperforms Support Vector Machine (SVM) and Decision Tree (DT) on most datasetsRandom forest proved superior for other evaluation metrics such as R^2, MAE, MSE, and RMSE. This shows that the random forest model is reliable and stable. In terms of making accurate predictions, this method is reliable. Although other models such as SVM and Decision Tree performed well on certain datasets, Random Forest was the most consistent of the models.

However, it is important to realize that the dataset's IT properties greatly impact how well these models function. For example, SVM and DT perform better than RF on datasets such as NASA 93, Deserhanis, and Cocomo 81, which have smaller effect sizes. This study emphasized a crucial discovery: no model is outperformed by the others on every dataset. Because of this, selecting a model needs to be customized for each dataset. Thus, there is no one-size-fits-all method for selecting models.

The stacking ensemble technique is useful when Random Forest is used as the core model because it can accommodate a wide range of data types. Random Forest's ability to generate more accurate predictions than Support Vector Machine (SVM) and Decision Tree algorithms further emphasizes the importance of selecting the appropriate base model for the ensemble. Overall, the stacking ensemble technique performs better and generates more dependable and consistent predictions across datasets by leveraging the strengths of Random Forest.

V. CONCLUSION

Accurate estimation of software development effort can make project planning successful and effective. But on the contrary, inaccurate estimation can lead to failure in project management. This research proves successful in effort estimation with the solution of applying stacking ensemble optimized by Bayesian hyperparameter tuning. This is achieved due to the utilization of meta learner in stacking and optimal parameter configuration in Bayesian. These results show how useful this method is in creating accurate software effort estimation tools and also provide ideas for further research.

The dataset used in this study may not be fully applicable to other types of projects with different characteristics, such as those with high complexity or limited data. As such, further evaluation is necessary to ensure the generalizability of the study's findings. For future research, it is recommended to employ diverse datasets to reinforce the conclusions drawn and to enhance the understanding of software effort estimation. Additionally, comparative analyses of various optimization strategies particularly for regression problems are essential. It is also critical to examine the effectiveness of different feature selection techniques, and to address issues related to scalability and computational complexity, especially when these approaches are applied to larger datasets within the domain of software effort estimation.

ACKNOWLEDGMENT

Authors would like to thank to Lembaga Pengelola Dana Pendidikan (LPDP) and Department of Informatics, Institut Teknologi Sepuluh Nopember (ITS), Surabaya, Indonesia, for supporting this research.

Conflicts of Interest: The authors declare no conflict of interest.

Data availability : The dataset used in the work are available at https://github.com/Derek-Jones/Software-estimation-datasets

Author Contributions: Muhammad Yusuf: Conceptualization, Methodology, Investigation, Writing – original draft, Data curation, Formal analysis, Visualization, Funding acquisition. Daniel Siahaan: Project administration, Resources, Writing – review & editing, Validation.

REFERENCES

[1] J. Leong, K. May Yee, O. Baitsegi, L. Palanisamy, and R. K. Ramasamy, "Hybrid Project Management between Traditional Software Development Lifecycle and Agile Based Product Development for Future Sustainability," *Sustainability*, vol. 15, no. 2, p. 1121, Jan. 2023, doi: 10.3390/su15021121.

[2] P. G. F. Matsubara, B. F. Gadelha, I. Steinmacher, and T. U. Conte, "SEXTAMT: A systematic map to navigate the wide seas of factors affecting expert judgment software estimates," *Journal of Systems and Software*, vol. 185, p. 111148, Mar. 2022, doi: 10.1016/j.jss.2021.111148.

[3] A.-E. Iordan, "An Optimized LSTM Neural Network for Accurate Estimation of Software Development Effort," *Mathematics*, vol. 12, no. 2, p. 200, Jan. 2024, doi: 10.3390/math12020200.

[4] K. Rathor, J. Kaur, U. A. Nayak, S. Kaliappan, R. Maranan, and V. Kalpana, "Technological Evaluation and Software Bug Training using Genetic Algorithm and Time Convolution Neural Network (GA-TCN)," in *2023 Second International Conference on Augmented Intelligence and Sustainable Systems (ICAISS)*, Trichy, India: IEEE, Aug. 2023, pp. 7–12. doi: 10.1109/ICAISS58487.2023.10250760.

[5] R. K. B. N and Y. Suresh, "Software Effort Estimation using ANN (Back Propagation)," in *2023 7th International Conference on Computing Methodologies and Communication (ICCMC)*, Erode, India: IEEE, Feb. 2023, pp. 1–2. doi: 10.1109/ICCMC56507.2023.10084264.

[6] Prisca Amajuoyi, Lucky Bamidele Benjamin, and Kudirat Bukola Adeusi, "Optimizing agile project management methodologies in high-tech software development," *GSC Adv. Res. Rev.*, vol. 19, no. 2, pp. 268–274, May 2024, doi: 10.30574/gscarr.2024.19.2.0182.

[7] B. Kapulica and K. Jurina, "Application of artificial intelligence in project management: analysis of potentials and challenges," *Et2er*, vol. 6, no. 2, pp. 115–121, Dec. 2024, doi: 10.70077/et2er.6.2.15.

[8] Ritu and P. Bhambri, "Enhancing software development effort estimation with a cloud-based data framework using use case points, fuzzy logic, and machine learning," *Discov Computing*, vol. 28, no. 1, p. 143, Jul. 2025, doi: 10.1007/s10791-025-09668-1.

[9] K. S. Thant and H. H. Khaung Tin, "LEARNING THE EFFICIENT ESTIMATION TECHNIQUES FOR SUCCESSFUL SOFTWARE PROJECT MANAGEMENT," *Innovare J Eng & Tech*, pp. 4–8, May 2023, doi: 10.22159/ijet.2023.v11i1.47605.

[10] B. A. Almahameed and M. Bisharah, "Applying Machine Learning and Particle Swarm Optimization for predictive modeling and cost optimization in construction project management," *Asian J Civ Eng*, vol. 25, no. 2, pp. 1281–1294, Feb. 2024, doi: 10.1007/s42107-023-00843-7.

[11] F. Sarro, R. Moussa, A. Petrozziello, and M. Harman, "Learning From Mistakes: Machine Learning Enhanced Human Expert Effort Estimates," *IIEEE Trans. Software Eng.*, vol. 48, no. 6, pp. 1868–1882, Jun. 2022, doi: 10.1109/TSE.2020.3040793.

[12] M. Fernandez-Diego, E. R. Mendez, F. Gonzalez-Ladron-De-Guevara, S. Abrahao, and E. Insfran, "An Update on Effort Estimation in Agile Software Development: A Systematic Literature Review," *IEEE Access*, vol. 8, pp. 166768–166800, 2020, doi: 10.1109/ACCESS.2020.3021664.

[13] A. Bahi, J. Gharib, and Y. Gahi, "Integrating Generative AI for Advancing Agile Software Development and Mitigating Project Management Challenges," *IJACSA*, vol. 15, no. 3, 2024, doi: 10.14569/IJACSA.2024.0150306.

[14] A. Mahardika and A. Retnowardhani, "Effectiveness Agile Project Management Tools For Managing Software Development Using F-AHP Method," in *2024 7th International Seminar on Research of Information Technology and Intelligent Systems (ISRITI)*, Yogyakarta, Indonesia: IEEE, Dec. 2024, pp. 860–865. doi: 10.1109/ISRITI64779.2024.10963563.

[15] J. O. Afape *et al.*, "Improving millimetre-wave path loss estimation using automated hyperparameter-tuned stacking ensemble regression machine learning," *Results in Engineering*, vol. 22, p. 102289, Jun. 2024, doi: 10.1016/j.rineng.2024.102289.

[16] J. Chen, J. Xu, S. Cai, X. Wang, H. Chen, and Z. Li, "Software Defect Prediction Approach Based on a Diversity Ensemble Combined With Neural Network," *IEEE Trans. Rel.*, vol. 73, no. 3, pp. 1487–1501, Sep. 2024, doi: 10.1109/TR.2024.3356515.

[17] B. R. P. Damoto *et al.*, "Deep Learning and Ensemble Approaches to Misinformation Detection in Digital News: A Systematic Review," in *2024 IEEE 10th Information Technology International Seminar (ITIS)*, Surabaya, Indonesia: IEEE, Nov. 2024, pp. 151–156. doi: 10.1109/ITIS64716.2024.10845709.

[18] Z. Sakhrawi, A. Sellami, and N. Bouassida, "Software Enhancement Effort Estimation using Machine Learning Regression Methods".

[19] Meharunnisa, M. Saqlain, M. Abid, M. Awais, and Ž. Stević, "Analysis of Software Effort Estimation by Machine Learning Techniques," *ISI*, vol. 28, no. 6, Dec. 2023, doi: 10.18280/isi.280602.

[20] D. K. Srivastava, A. K. Sharma, and D. Choudhary, "Software Development Effort Estimation Using Machine Learning Techniques: Multi-linear Regression versus Random Forest," in *2021 International Conference on Computing, Communication and Green Engineering (CCGE)*, Pune, India: IEEE, Sep. 2021, pp. 1–5. doi: 10.1109/CCGE50943.2021.9776394.

[21] B. R. Paradisiaca Darnoto, D. Siahaan, and D. Purwitasari, "A Comprehensive Ensemble Deep Learning Method for Identifying Native Advertising in News Articles," in *2023 IEEE 8th International Conference On Software Engineering and Computer Systems (ICSECS)*, Penang, Malaysia: IEEE, Aug. 2023, pp. 164–169. doi: 10.1109/ICSECS58457.2023.10256392.

[22] I.-G. Chelaru, "Enhancing the performance of software effort estimation through boosting ensemble learning," in *2023 25th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, Nancy, France: IEEE, Sep. 2023, pp. 300–307. doi: 10.1109/SYNASC61333.2023.00051.

[23] M. Yusuf, A. Haq, and S. Rochimah, "Integrating Adaptive Sampling with Ensembles Model for Software Defect Prediction," *KINETIK*, May 2025, doi: 10.22219/kinetik.v10i2.2191.

[24] M. Hammad and M. Amin, "Assuring Software Reuse Success Using Ensemble MachineLearning Algorithms," *IJCDS*, vol. 13, no. 1, pp. 69–81, Jan. 2023, doi: 10.12785/ijcds/130107.

[25] Zainab Rustum Mohsin, "Investigating the Use of an Adaptive Neuro-Fuzzy InferenceSystem in Software Development Effort Estimation," *ijcsm*, pp. 18–24, Jul. 2021, doi: 10.52866/ijcsm.2021.02.02.003.

[26] M. G. Meharie, W. J. Mengesha, Z. A. Gariy, and R. N. N. Mutuku, "Application of stacking ensemble machine learning algorithm in predicting the cost of highway construction projects," *ECAM*, vol. 29, no. 7, pp. 2836–2853, Aug. 2022, doi: 10.1108/ECAM-02-2020-0128.

[27] H. Cho, Y. Kim, E. Lee, D. Choi, Y. Lee, and W. Rhee, "Basic Enhancement Strategies When Using Bayesian Optimization for Hyperparameter Tuning of Deep Neural Networks," *IEEE Access*, vol. 8, pp. 52588–52608, 2020, doi: 10.1109/ACCESS.2020.2981072.

[28] S. Hameed, Y. Elsheikh, and M. Azzeh, "An optimized case-based software project effort estimation using genetic algorithm," *Information and Software Technology*, vol. 153, p. 107088, Jan. 2023, doi: 10.1016/j.infsof.2022.107088.

[29] A. Jadhav and S. K. Shandilya, "Reliable machine learning models for estimating effective software development efforts: A comparative analysis," *Journal of Engineering Research*, vol. 11, no. 4, pp. 362–376, Dec. 2023, doi: 10.1016/j.jer.2023.100150.

[30] Z. Sakhrawi, T. Labidi, A. Sellami, and N. Bouassida, "A Stacking Ensemble Learning Model for Software Development Cost Estimation".

[31] A. Kaushik, K. Sheoran, R. Kapur, N. Bhutani, B. Singh, and H. Sharma, "SENSE: software effort estimation using novel stacking ensemble learning," *Innovations Syst Softw Eng*, vol. 21, no. 2, pp. 769–785, Jun. 2025, doi: 10.1007/s11334-024-00581-2.

[32] P. Srivastava, N. Srivastava, R. Agarwal, and P. Singh, "NEMAEP: A NOVEL ENSEMBLE MACHINE LEARNING FRAMEWORK FOR ACCURATE EFFORT ESTIMATION IN SOFTWARE PROJECTS," . *Vol.*, no. 24.

[33] S. Kassaymeh, M. Alweshah, M. A. Al-Betar, A. I. Hammouri, and M. A. Al-Ma'aitah, "Software effort estimation modeling and fully connected artificial neural network optimization using soft computing techniques," *Cluster Comput*, vol. 27, no. 1, pp. 737–760, Feb. 2024, doi: 10.1007/s10586-023-03979-y.