SELI: The Stacking-Blending Ensemble Learning and Interpretability Framework for Software Effort Estimation

Puguh Jayadi
Department of Electrical Engineering and Informatics, Faculty of Engineering, State University of Malang, Malang, Indonesia | Department of Informatics, Faculty of Engineering, PGRI University of Madiun, Indonesia
puguh.jayadi.2405349@students.um.ac.id

Didik Dwi Prasetya
Department of Electrical Engineering and Informatics, Faculty of Engineering, State University of Malang, Malang, Indonesia
didikdwi@um.ac.id (corresponding author)

Triyanna Widiyaningtyas
Department of Electrical Engineering and Informatics, Faculty of Engineering, State University of Malang, Malang, Indonesia
triyannaw.ft@um.ac.id

Azlan Mohd Zain
Faculty of Computing, Universiti Teknologi Malaysia, Johor Bahru, Malaysia
azlanmz@utm.my

Received: 23 October 2025 | Revised: 9 December 2025 and 23 December 2025 | Accepted: 24 December 2025

 DOI: https://doi.org/10.48084/etasr.15742

ABSTRACT

Software Effort Estimation (SEE) is an important element in software engineering that predicts the time, cost, and human resources needed for project development. Accurate estimates play a vital role in project planning and risk control, but many existing models still struggle to balance prediction accuracy and interpretability. Machine learning and deep learning–based approaches, such as SENSE, NIVIM, Mdb+MoWE, and GGSNN, can improve accuracy, but they are complex and challenging to explain. This study proposes the Stacking-Blending Ensemble Learning and Interpretability Framework (SELI), a lightweight, adaptive SEE framework integrated with Explainable Artificial Intelligence (XAI) to address these issues. SELI uses ridge-based meta-learning and non-negative blending to improve stability across datasets and integrates interpretability methods, such as Shapley Additive Explanations (SHAP) and Local Interpretable Model-agnostic Explanations (LIME), to provide insight into feature contributions. The results of experiments on six benchmark datasets show that SELI achieves Mean Absolute Error (MAE) values of 0.0042–0.0577 and coefficients of determination (R^2) of up to 0.9939, outperforming the baseline model. XAI analysis reveals that SELI can provide consistent and transparent explanations of the factors influencing estimates. These findings indicate that SELI is an important step in developing an accurate SEE model and has great potential to support decision-making in software project management.

Keywords-Software Effort Estimation (SEE); stacking-blending ensemble learning; Explainable Artificial Intelligence (XAI); ridge-based meta-learning

I. INTRODUCTION

Software Effort Estimation (SEE) is an important field in software engineering that estimates the resources, time, and costs required for project development [1]. SEE plays a strategic role in project management because accurate estimates help reduce the risk of delays and cost overruns [2]. A comprehensive review has shown that improving accuracy remains an open research topic in SEE [3]. Over the past decades, approaches from classical SEE models such as

COCOMO [4] and Function Point Analysis [5] to modern machine learning–based refinements have evolved to improve cost estimation [6]. Machine learning techniques such as Linear Regression [7] and Artificial Neural Networks [8] have been extensively applied in SEE. Several studies have applied metaheuristic algorithms, such as Particle Swarm Optimization [9], Grey Wolf Optimization [10], and the Firefly algorithm [11], to improve SEE models. However, achieving a balance between estimation accuracy and model interpretability remains a challenge in contemporary SEE research.

Recent studies have explored deep learning architectures for effort estimation, where hybrid models such as Deep Learning Neural Networks [12] and Graph-Based Representations [13] provide improved predictive power while maintaining task-specific interpretability. Ensemble methods such as Random Forest [14] and Gradient Boosting [15] have proven effective for nonlinear SEE problems, with recent adaptations achieving robust early-stage estimation performance [16]. However, these ensemble models still exhibit limited cross-dataset stability and generalization capacity. This narrows the research area towards solutions that are not only accurate but also adaptive and interpretable, allowing project managers and stakeholders to apply the estimation results practically.

Previous research results used as baselines, such as GGSNN [13], Mdb+MoWE [16], SENSE [17], NIVIM [18], LSTM-Jaiswal [19], RF-Yusuf [20] and HTRR-RNN [21], show performance differences across datasets. Most of these approaches prioritize accuracy optimization while providing limited insight into the drivers of the estimates. This leaves two persistent gaps in SEE research: (i) highly accurate models often remain black boxes and (ii) models that are easier to interpret frequently sacrifice accuracy and computational efficiency (runtime).

To bridge this gap, this study proposes the Stacking-Blending Ensemble Learning and Interpretability Framework (SELI) as a new approach that combines the strengths of stacking and blending with adaptive mechanisms and the integration of Explainable Artificial Intelligence (XAI) such as Shapley Additive Explanations (SHAP) and Local Interpretable Model-agnostic Explanations (LIME) [22]. SELI is designed as a framework that can adapt the ensemble structure to dataset characteristics. With these innovations, SELI is expected to be an effective solution to the need for accurate and transparently explainable SEE models.

II. METHODOLOGY

All details contained in the methodology are illustrated in Figure 1. The initial stages focused on data preprocessing to ensure the dataset quality and prevent bias. The datasets consist of UCP [23], Maxwell [24], Albrecht [25], COCOMO81 [4], Desharnais [26], and China [27], with statistical descriptions shown in Table I.

Each dataset was cleaned of duplicates and missing values to maintain data reliability and minimize technical debt accumulation during preprocessing [28]. Imputation was then performed using ensemble-based methods, such as median and mode, which are effective in SEE [29]. Several resampling methods adapted from the imbalanced learning literature [30] were evaluated to mitigate skewed distributions. All numerical variables were normalized to the range 0–1 to ensure consistent feature scaling and better convergence during learning [31]. Feature prioritization strategies were also implemented to support feature scaling alignment [32]. To mitigate the influence of extreme outliers, noise filtering and resampling were applied before winsorizing at the upper and lower percentile limits [33]. After that, the data were divided into three subsets: training, validation, and testing [34]. This division used the anti-leakage principle, in which normalization parameters were calculated solely from the training data to avoid bias [31].

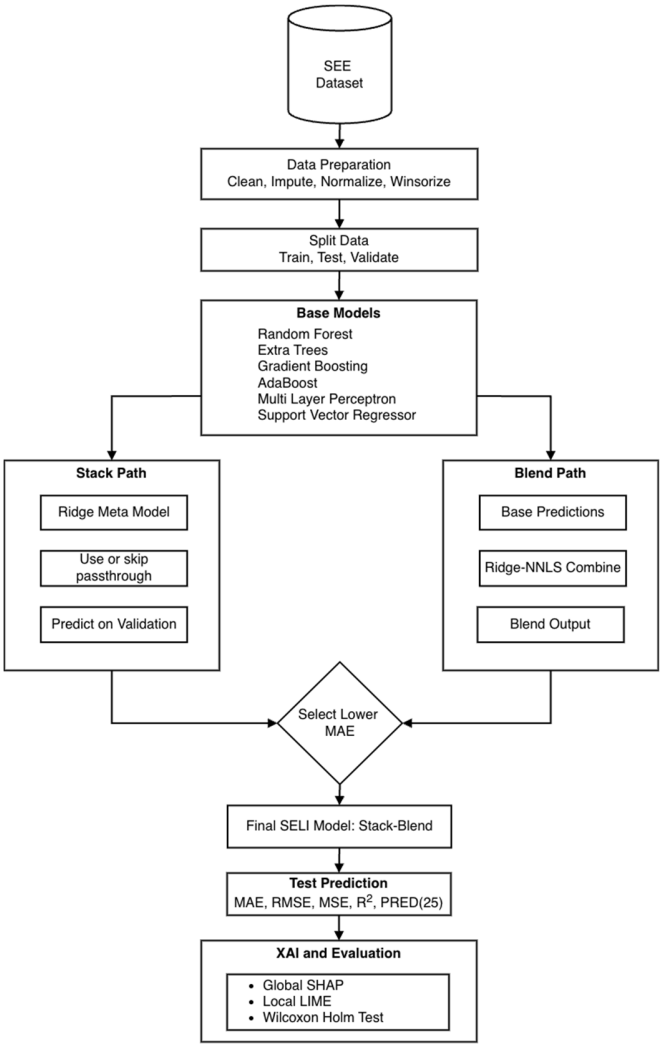

Fig. 1. SELI methodology.

The next stage is central processing, which is the core of the proposed framework. SELI uses six heterogeneous basic learning models. These models were selected to ensure algorithmic diversity by covering the main learning groups of bagging/tree ensembles (Random Forest, Extra Trees), boosting ensembles (Gradient Boosting, AdaBoost), neural

networks (MLP), and Support Vector Regressor (SVR) [36]. Such heterogeneity is crucial in stacking-blending because it allows the ensemble to capture both linear and nonlinear relationships in the SEE datasets and reduce correlated errors among the base predictors, enabling the meta-learner to combine complementary strengths effectively [35].

TABLE I. DESCRIPTIVE AND STATISTICAL DETAILS OF THE DATASETS

Dataset	#Proj	#Features	Min	Max	Mean	Skewness	Kurtosis
Albrecht [25]	24	7	0.5	105	21.87	2.30	4.70
China [27]	499	18	26	54,620	3,921.04	3.92	19.30
COCOMO81 [4]	63	17	5.9	11,400	683.30	4.26	19.35
Desharnais [26]	77	11	546	23,940	4,833.90	2.03	5.30
Maxwell [24]	62	26	583	63,694	8,223.21	3.18	12.02
UCP [23]	71	15	5,775	7,970	6,558.72	0.57	-0.92

SELI uses two ensemble paths that are evaluated in parallel. In the stacking path, ridge regression acts as a meta-learner, passing through the original features to control complexity and maintain generalization. In the blending path, all base model predictions are combined using Ridge-Non-Negative Least Squares (Ridge-NNLS), which applies non-negative weights to remain stable across scale variations. Since stacking can achieve higher accuracy on some datasets, whereas blending is more stable across certain data structures, SELI selects the path with the lowest validation Mean Absolute Error (MAE). The best path is set as the final model.

Evaluation metrics such as MAE, Root Mean Square Error (RMSE), Mean Square Error (MSE), coefficient of determination (R^2), and Predictive Accuracy within 25% (PRED(25)) were used to assess predictive performance [35]. The formulas are presented in (1)–(5). The MAE metric calculates the average absolute difference between actual and predicted values. The RMSE metric highlights errors by using the square root of the mean square difference. Meanwhile, the MSE is the value before the root. To measure the model's ability to explain data variation, R^2 is used, and PRED(25) quantifies the proportion of predictions whose relative errors are less than 25% of the actual effort.

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

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

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

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

$$\mathrm{PRED}(25) = \frac{1}{n}\sum_{i=1}^{n}\left[\frac{|y_i - \hat{y}_i|}{y_i} < 0.25\right] \quad (5)$$

The postprocessing stage introduces interpretability to support decision-making in SEE. SHAP [37] (6) provides global insights that show which factors consistently influence the prediction across datasets. These insights help project managers understand the main cost drivers and adjust planning, resource allocation, and risk management. LIME [38] (7) provides specific explanations for each project so that users can assess the reasons behind specific estimates. This approach makes SELI outputs easier to apply in real project management contexts. SHAP and LIME provide different and complementary types of explanations [39]. SHAP provides a consistent global picture of each feature's contribution, enabling model results to be understood. LIME provides explanations at the individual level so that the reasoning behind a single prediction can be evaluated directly. Combining these two methods yields a complete interpretation, as users can understand the model's general patterns and verify each estimate.

$$\phi_i = \sum_{S \subseteq F \setminus \{i\}} \frac{|S|!(|F|-|S|-1)!}{|F|!}[f(S \cup \{i\}) - f(S)] \quad (6)$$

$$\hat{f}(x) = \beta_0 + \sum_{i=1}^{p} \beta_i x_i \quad (7)$$

The prediction results from the final SELI model are evaluated using the Wilcoxon signed-rank test [40] with Holm correction [41] and Cliff's δ effect size [42], following comparative procedures in SEE evaluation studies [16]. The Wilcoxon signed-rank test was used because the SEE performance metrics were not normally distributed, as confirmed by the Shapiro-Wilk test ($p < 0.05$), making it appropriate for paired and nonparametric comparisons across the entire dataset. Holm's correction was applied to control the Type I error rate when multiple paired statistical tests were performed. Additionally, Cliff's δ effect size was reported to measure the magnitude of performance differences and to assess their practical relevance in SEE. For error-based metrics such as MAE and RMSE, negative values of Cliff's δ indicate lower estimation errors. In contrast, for goodness-of-fit metrics such as R^2 and PRED(25), positive Cliff's δ values indicate better estimation performance.

III. RESULTS AND DISCUSSION

The best performance achieved by SELI across all datasets is shown in Table II. The values in bold are the best values, indicating the highest performance among all models tested. In Albrecht, the MAE value of 0.0204 and R^2 of 0.9703 indicate high accuracy. The China dataset shows superior performance, with an MAE of 0.0042 and an R^2 of 0.9816. In COCOMO81, SELI achieved R^2 of 0.9939, far exceeding the baseline. Desharnais, Maxwell, and UCP each show model stability with R^2 values of 0.8486, 0.7805, and 0.9374, respectively. The PRED(25) value reaches a maximum of 1.000 on UCP, confirming the reliability of SELI's estimation.

Figure 2 presents the average rank, defined as the mean position of each model across all datasets and evaluation metrics. The rank is computed by ordering models by the five SEE metrics (MAE, RMSE, MSE, R^2, and PRED(25)), then averaging their positions across all datasets and metrics. Lower average ranks indicate better overall performance, meaning the model more consistently achieves top results across different metrics and datasets. As shown, SELI attains the best average rank (1.0) and reflects the most consistent performance. RF-Yusuf and Mdb+MoWE occupy mid-level ranks (2.0), whereas

HTRR-RNN and GGSNN show the weakest overall performance with ranks above 3.5. These rankings align with the statistical tests, confirming that SELI delivers both high accuracy and stable performance across heterogeneous software project datasets.

TABLE II. EVALUATION METRICS FOR ALL DATASETS

Dataset	Model	MAE	RMSE	MSE	R^2	PRED(25)
Albrecht [25]	SELI (proposed)	**0.0204**	**0.0240**	**0.0108**	**0.9703**	**0.9800**
	SENSE [17]	0.0300	0.0329	–	0.9037	–
	NIVIM [18]	0.0254	0.0831	–	0.9131	–
	Mdb+MoWE [16]	0.0670	0.0960	–	–	0.8800
China [27]	SELI (proposed)	**0.0042**	**0.0102**	**0.0002**	**0.9816**	**0.9900**
	SENSE [17]	0.0438	0.0125	–	0.9391	–
	RF-Yusuf [20]	0.0058	–	–	0.7725	–
	HTRR-RNN [21]	0.1800	0.3800	–	–	–
	Mdb+MoWE [16]	0.0110	0.0210	–	–	0.8000
COCOMO81 [4]	SELI (proposed)	**0.0046**	**0.0156**	**0.0013**	**0.9939**	**0.8462**
	GGSNN [13]	0.8439	0.8850	–	–	–
	NIVIM [18]	0.0498	0.1013	–	0.6032	–
	Mdb+MoWE [16]	0.0060	0.0210	–	–	0.7800
Desharnais [26]	SELI (proposed)	**0.0577**	**0.0874**	**0.0166**	**0.8486**	**0.9750**
	LSTM-Jaiswal [19]	0.2606	0.0968	0.6296	0.7934	–
	NIVIM [18]	0.0699	0.1134	–	0.6432	–
	Mdb+MoWE [16]	0.0660	0.0960	–	–	0.9500
Maxwell [24]	SELI (proposed)	**0.0355**	**0.0175**	**0.0003**	**0.7805**	**0.9769**
	SENSE [17]	0.0406	0.0487	–	0.3199	–
	LSTM-Jaiswal [19]	0.0619	0.0285	0.0008	0.7299	–
	Mdb+MoWE [16]	0.0400	0.1250	–	–	0.8900
UCP [23]	SELI (proposed)	**0.0282**	**0.0325**	**0.0011**	**0.9374**	**1.0000**
	GGSNN [13]	0.7171	0.8313	–	–	–
	NIVIM [18]	0.0398	0.0812	–	0.7892	–

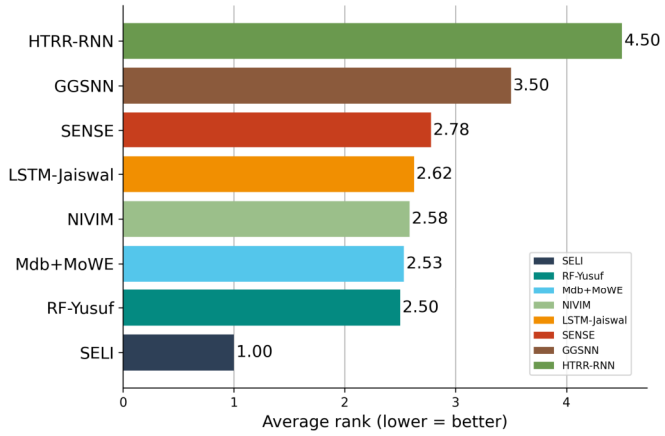

Fig. 2. Average rank of all models across datasets based on five SEE metrics (MAE, RMSE, MSE, R2, and PRED(25)).

Figure 3 illustrates the inter-metric correlations among MAE, RMSE, MSE, R^2, and PRED(25). MAE, RMSE, and MSE show strong positive correlations ($r > 0.9$), indicating consistent performance patterns, whereas a negative correlation with R^2 confirms higher generalization at lower error magnitudes. The moderate positive relation between PRED(25) and the error-based measures reflects its partial dependence on trends in absolute deviation. Overall, this relationship confirms that reductions in MAE, RMSE, and MSE parallel improvements in the SELI framework's predictive reliability and generalization capability.

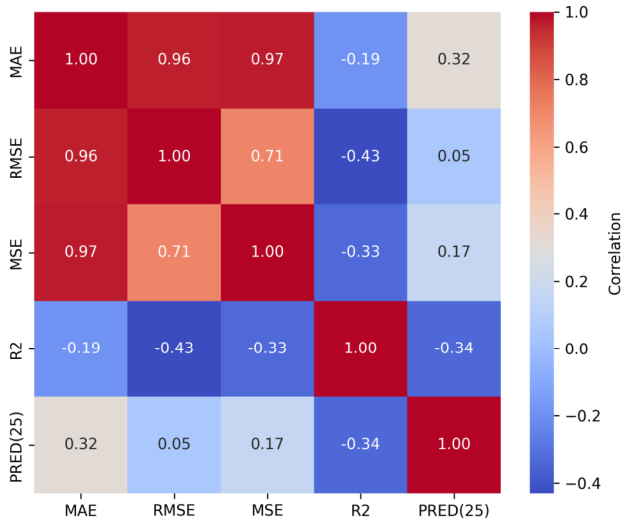

Fig. 3. Correlation between performance metrics.

Figure 4 shows the percentage increase in SELI performance compared to the previous best model for each dataset. The MAE reductions are observed on UCP (29%) and China (28%), followed by the COCOMO81 (24%) and Albrecht (19%) datasets. Overall, SELI reduces MAE by 11%–29% across datasets, indicating stable gains under different project characteristics.

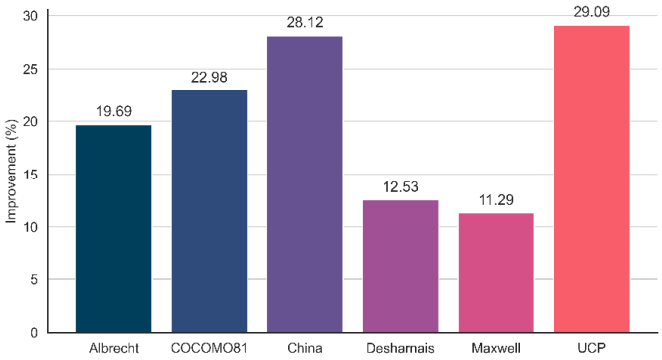

Fig. 4. SELI improvement compared with the best prior model (MAE-based).

Figure 5 shows that SELI consistently maintains the lowest runtime (2–3 s) across all datasets, confirming its computational efficiency. Only three baselines, namely HTRR-

RNN, GGSNN, and Mdb+MoWE, reported runtime, and the reporting did not cover all datasets. This condition emphasizes that SELI's efficiency is more clearly evident because it is displayed consistently across all test scenarios.

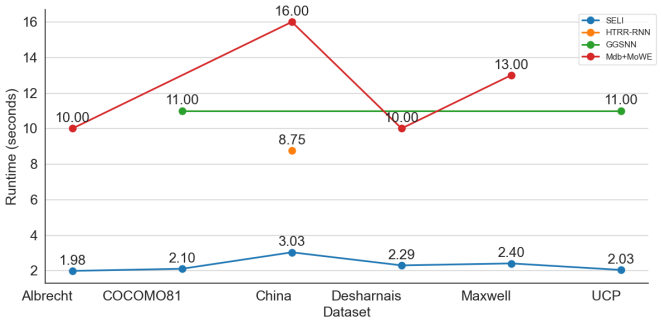

Fig. 5. Runtime comparison of SELI with other baselines.

The functional characteristics of applications and development contexts were identified as the top 10 strongest determinants of effort in SELI across several datasets, as shown in Table III. The dominance of app_InfServ (1492) and app_CustServ (856) indicates that application type greatly influences effort patterns. The contribution of dev_mode_embedded (426) and dev_mode_semidetached (304) confirms that the complexity of the development process and environment also shapes model decisions. Interface attributes and programming languages also have an impact.

TABLE III. GLOBAL LIME TOP 10 FEATURES

Feature	Detail	Dataset	AbsLocalWeightSum
app_InfServ	Application type: Infrastructure Service	Maxwell [24]	1,492.28
app_CustServ	Application type: Customer Service	Maxwell [24]	856.07
dev_mode_embedded	Embedded development mode	COCOMO 81 [4]	425.63
app_TransPro	Application type: Transaction Processing	Maxwell [24]	386.45
dev_mode_semidetached	Semidetached development mode	COCOMO 81 [4]	304.34
app_MIS	Application type: MIS	Maxwell [24]	249.76
ifc_GUI	Graphics in user interface technology are used	Maxwell [24]	160.26
ifc_TextUI	Text on the user interface technology used	Maxwell [24]	160.26
nlan	Number of programming languages being used	Maxwell [24]	146.72
t13	Staff application knowledge	Maxwell [24]	124.77

The visual in Figure 6 confirms that SELI identifies the dominance of application types and development modes, produces consistent results across multiple datasets, and is followed by software development techniques such as programming languages and user interfaces. SELI results demonstrate stability in identifying local weights and in capturing project determinants in greater detail.

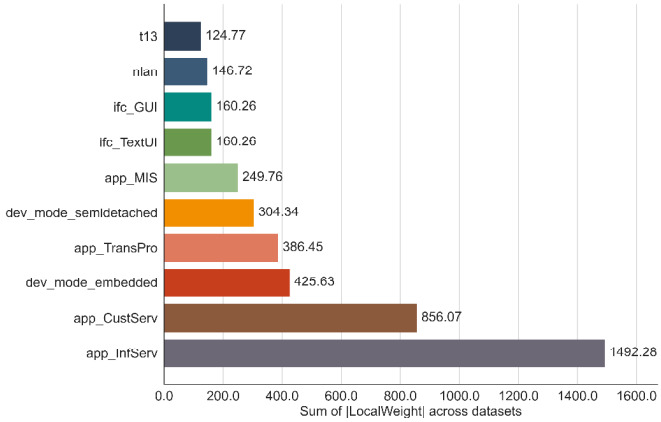

Fig. 6. Global LIME bar (top 10 predictor features).

The heatmap in Figure 7 clarifies the distribution of feature influence across datasets. The intensity pattern shows that application-type features are very dominant in Maxwell, whereas development mode is more influential in COCOMO81. This variation confirms SELI's ability to adjust its explanatory weights to the characteristics of each dataset.

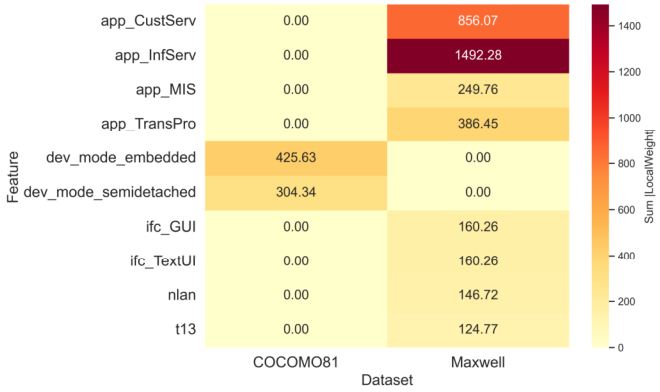

Fig. 7. Global LIME heatmap (top 10 features × datasets).

The global SHAP results in Table IV show that project functional measures such as AFP (2,506.14), Added (2,392.13), Duration (1,725.52), and Input (1,438.07) contribute significantly to SELI predictions, especially in the China dataset. Quality and process factors, such as demand volatility and tool usage in Maxwell, are additional determinants of effort estimates.

Figure 8 clarifies the global SHAP contribution pattern by showing the consistent dominance of functional features, such as size and functional components, in SELI predictions. This visualization validates the previous table by showing a stable distribution of contributions across features, while confirming that process variables have only a secondary influence compared to the project's structural characteristics. The heatmap in Figure 9 shows the largest SHAP contribution concentrations in the China dataset for functional-size features, whereas process features are more dominant in the Maxwell dataset. This pattern confirms SELI's adaptability in adjusting feature sensitivity to different dataset characteristics.

TABLE IV. GLOBAL SHAP TOP 10 FEATURES

Feature	Detail	Dataset	AbsContributionSum
AFP	Adjusted function points	China [27]	2,506.14
Added	Function points of added functions	China [27]	2,392.13
Duration	Total elapsed time for the project	China [27]	1,725.52
Input	Function points of input	China [27]	1,438.07
t08	Requirements volatility	Maxwell [24]	1,323.47
data	Database size	China [27]	1,266.45
Interface	Function points of the external interface added	China [27]	1,043.98
File	Function points of internal logical files	China [27]	913.80
t11	Installation requirements	Maxwell [24]	791.32
t06	Tools use	Maxwell [24]	742.66

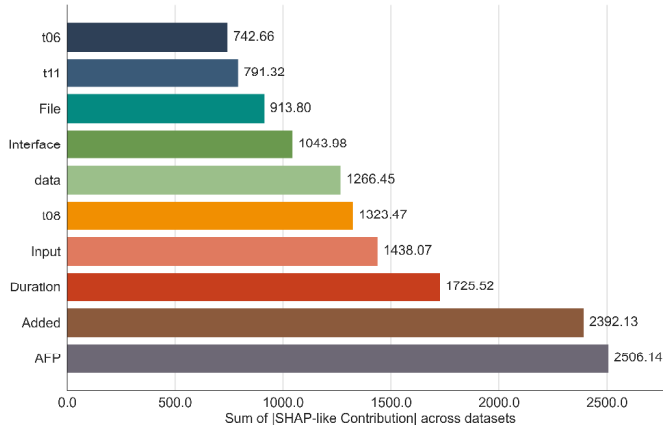

Fig. 8. Global SHAP bar (top 10 predictor features).

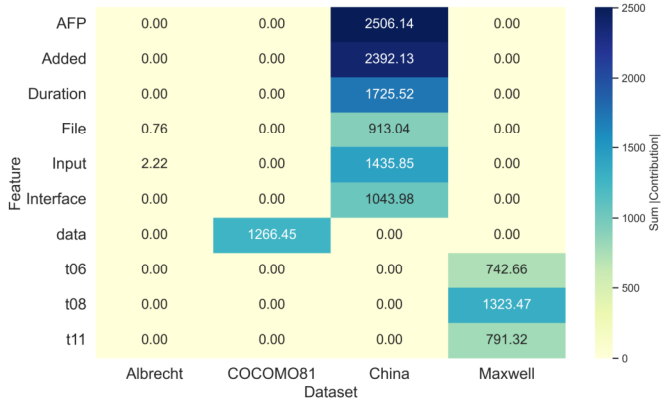

Fig. 9. Global SHAP heatmap (top 10 features × datasets).

Table V presents the results of the Wilcoxon signed-rank test with Holm correction and Cliff's δ effect size comparing SELI with three baseline models. For error-based metrics, negative Cliff's δ values indicate that SELI tends to produce lower estimation errors than Mdb+MoWE (-0.44), SENSE (-0.777), and NIVIM (-0.5), with effect sizes ranging from moderate to large. For goodness-of-fit metrics, positive Cliff's δ values reflect improvements in explanatory and predictive accuracy. The mean MAE of SELI ranges from 0.0204 to 0.027, compared with baseline values between 0.038 and 0.046, indicating a measurable reduction in estimation error. Some pairwise comparisons do not reach statistical significance, likely due to the limited number of datasets and the conservative nature of the Holm correction.

TABLE V. WILCOXON SIGNED-RANK TEST WITH HOLM AND CLIFF

Metric	Baseline	n_pairs	SELI_mean	BASE_mean	Cliff's δ	Effect size	p_holm
MAE	Mdb+MoWE	5	0.02	0.04	-0.44	Medium	0.09
MAE	NIVIM	4	0.03	0.05	-0.5	Large	0.13
MAE	SENSE	3	0.02	0.04	-0.78	Large	0.13
RMSE	Mdb+MoWE	5	0.03	0.07	-0.68	Large	0.09
RMSE	NIVIM	4	0.04	0.09	-0.75	Large	0.13
RMSE	SENSE	3	0.02	0.03	-0.56	Large	0.13
R^2	NIVIM	4	0.94	0.72	0.88	Large	0.13
R^2	SENSE	3	0.91	0.72	0.56	Large	0.13
PRED(25)	Mdb+MoWE	5	0.95	0.86	0.76	Large	0.03

Although the experimental results show a clear performance advantage, several potential threats to validity should be acknowledged. The limited number of datasets may affect the generalizability of the results, particularly for industrial-scale projects. Dataset imbalance and small sample sizes in some benchmarks could also introduce bias. To mitigate this risk, multi-level validation and replication are necessary. Future research should test SELI on real-world corporate data and in cross-organizational contexts to assess its external validity.

IV. CONCLUSION

This study focuses on developing the Stacking-Blending Ensemble Learning and Interpretability Framework (SELI) as a novel approach to Software Effort Estimation (SEE) that produces more accurate and interpretable estimates. By integrating adaptive stacking and blending methods with Explainable Artificial Intelligence (XAI), SELI was tested on six datasets. The test results show that SELI excels across all key metrics and achieves an accuracy improvement compared to current models such as SENSE, NIVIM, and Mdb+MoWE. Interpretability analyses using Shapley Additive Explanations (SHAP) and Local Interpretable Model-Agnostic Explanations (LIME) confirm that SELI can identify dominant features, thereby enabling transparent explanations of the estimation results.

The results of this study show that SELI can improve prediction performance and provide a clear understanding of the factors that influence effort estimation. For further research, it is recommended to explore automatic meta-learning optimization to improve adaptivity and to apply SELI in real industrial domains to strengthen the model's external validity. Integration with interactive XAI visualization techniques is also recommended to strengthen the acceptance and utilization

of SELI in modern software engineering practices. SELI bridges the long-standing gap between accuracy and interpretability in SEE, aligning with trustworthy Artificial Intelligence (AI) principles for modern project management.

REFERENCES

[1] P. Manchala and M. Bisi, "TSoptEE: two-stage optimization technique for software development effort estimation," *Cluster Computing*, vol. 27, no. 7, pp. 8889–8908, Oct. 2024, https://doi.org/10.1007/s10586-024-04418-2.

[2] X. Shen, J. Lu, S. Li, and L. Song, "Feature mapping based on heterogeneous cross-company effort estimation," *Software Quality Journal*, vol. 32, no. 4, pp. 1717–1761, Dec. 2024, https://doi.org/10.1007/s11219-024-09697-x.

[3] C. H. Rashid *et al.*, "Software Cost and Effort Estimation: Current Approaches and Future Trends," *IEEE Access*, vol. 11, pp. 99268–99288, 2023, https://doi.org/10.1109/ACCESS.2023.3312716.

[4] B. W. Boehm, "Software Engineering Economics," *IEEE Transactions on Software Engineering*, vol. SE-10, no. 1, pp. 4–21, Jan. 1984, https://doi.org/10.1109/TSE.1984.5010193.

[5] A. J. Albrecht, "Measuring Application Development Productivity," in *Proceedings of IBM Applications Development Symposium*, Monterey, CA, USA, 1979, pp. 83–92.

[6] C. H. Rashid, I. Shafi, B. H. A. Khattak, M. Safran, S. Alfarhood, and I. Ashraf, "ANN-based software cost estimation with input from COCOMO: CANN model," *Alexandria Engineering Journal*, vol. 113, pp. 681–694, Feb. 2025, https://doi.org/10.1016/j.aej.2024.11.042.

[7] S. Wakurdekar *et al.*, "Novel Approach to Design a Model for Software Effort Estimation Using Linear Regression," *Journal of Electrical Systems*, vol. 20, no. 2, pp. 2306–2315, Apr. 2024, https://doi.org/10.52783/jes.1997.

[8] D. Rankovic, N. Rankovic, M. Ivanovic, and L. Lazic, "Convergence rate of Artificial Neural Networks for estimation in software development projects," *Information and Software Technology*, vol. 138, Oct. 2021, Art. no. 106627, https://doi.org/10.1016/j.infsof.2021.106627.

[9] M. M. Draz, O. Emam, and S. M. Azzam, "Software cost estimation predication using a convolutional neural network and particle swarm optimization algorithm," *Scientific Reports*, vol. 14, no. 1, June 2024, Art. no. 13129, https://doi.org/10.1038/s41598-024-63025-8.

[10] N. M. Alsheikh and N. M. Munassar, "Improving Software Effort Estimation Models Using Grey Wolf Optimization Algorithm," *IEEE Access*, vol. 11, pp. 143549–143579, 2023, https://doi.org/10.1109/ACCESS.2023.3340140.

[11] A. J. AlMutlaq, D. N. A. Jawawi, and A. F. B. Arbain, "Weight Optimization Based on Firefly Algorithm for Analogy-based Effort Estimation," *International Journal of Advanced Computer Science and Applications*, vol. 14, no. 6, pp. 617–628, June 2023, https://doi.org/10.14569/IJACSA.2023.0140666.

[12] M. R. Devadas and P. Samuel, "Enhancing effort estimation in global software development using a unique combination of Neuro Fuzzy Logic and Deep Learning Neural Networks (NFDLNN)," *Network: Computation in Neural Systems*, vol. 36, no. 4, pp. 1606–1626, Oct. 2025, https://doi.org/10.1080/0954898X.2024.2376703.

[13] N. Rankovic, D. Rankovic, M. Ivanovic, and J. Kaljevic, "Interpretable software estimation with graph neural networks and orthogonal array tunning method," *Information Processing & Management*, vol. 61, no. 5, Sept. 2024, Art. no. 103778, https://doi.org/10.1016/j.ipm.2024.103778.

[14] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, Oct. 2001, https://doi.org/10.1023/A:1010933404324.

[15] J. H. Friedman, "Greedy function approximation: A gradient boosting machine.," *The Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, Oct. 2001, https://doi.org/10.1214/aos/1013203451.

[16] A. Yasmin, W. H. Butt, and A. Daud, "Ensemble effort estimation with metaheuristic hyperparameters and weight optimization for achieving accuracy," *Plos One*, vol. 19, no. 4, Apr. 2024, Art. no. e0300296, https://doi.org/10.1371/journal.pone.0300296.

[17] A. Kaushik, K. Sheoran, R. Kapur, N. Bhutani, B. Singh, and H. Sharma, "SENSE: software effort estimation using novel stacking ensemble learning," *Innovations in Systems and Software Engineering*, vol. 21, no. 2, pp. 769–785, June 2025, https://doi.org/10.1007/s11334-024-00581-2.

[18] S. S. Ali, J. Ren, and J. Wu, "Framework to improve software effort estimation accuracy using novel ensemble rule," *Journal of King Saud University - Computer and Information Sciences*, vol. 36, no. 9, Nov. 2024, Art. no. 102189, https://doi.org/10.1016/j.jksuci.2024.102189.

[19] A. Jaiswal, J. Raikwal, and P. Raikwal, "Using Deep Learning Model to Estimate Cost of Software Project Development," *International Journal of Engineering Trends and Technology*, vol. 73, no. 5, pp. 369–382, May 2025, https://doi.org/10.14445/22315381/IJETT-V73I5P130.

[20] M. Yusuf and D. Siahaan, "Software Effort Estimation Using Stacking Ensemble and Bayesian Optimization," *Journal of Applied Science and Technology Trends*, vol. 6, no. 2, pp. 161–168, Aug. 2025, https://doi.org/10.38094/jastt62310.

[21] K. Harish Kumar and K. Srinivas, "An improved analogy-rule based software effort estimation using HTRR-RNN in software project management," *Expert Systems with Applications*, vol. 251, Oct. 2024, Art. no. 124107, https://doi.org/10.1016/j.eswa.2024.124107.

[22] A. Barredo Arrieta *et al.*, "Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI," *Information Fusion*, vol. 58, pp. 82–115, June 2020, https://doi.org/10.1016/j.inffus.2019.12.012.

[23] R. Silhavy, P. Silhavy, and Z. Prokopova, "Analysis and selection of a regression model for the Use Case Points method using a stepwise approach," *Journal of Systems and Software*, vol. 125, pp. 1–14, Mar. 2017, https://doi.org/10.1016/j.jss.2016.11.029.

[24] K. D. Maxwell and P. Forselius, "Benchmarking software development productivity," *IEEE Software*, vol. 17, no. 1, pp. 80–88, Jan. 2000, https://doi.org/10.1109/52.820015.

[25] A. J. Albrecht and J. E. Gaffney, "Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation," *IEEE Transactions on Software Engineering*, vol. SE-9, no. 6, pp. 639–648, Nov. 1983, https://doi.org/10.1109/TSE.1983.235271.

[26] J. M. Desharnais, "Adjustment Model for Function Point Scope Factors–A Statistical Study," in *IFPUG Spring Conference*, Florida, USA, 1990.

[27] T. Menzies *et al.*, "Local versus Global Lessons for Defect Prediction and Effort Estimation," *IEEE Transactions on Software Engineering*, vol. 39, no. 6, pp. 822–834, June 2013, https://doi.org/10.1109/TSE.2012.83.

[28] J. Wiriyakul and T. Senivongse, "Estimating Technical Debt Through Changes Related to Code Smells in the Evolution of Software Projects," *IEEE Access*, vol. 13, pp. 107130–107153, 2025, https://doi.org/10.1109/ACCESS.2025.3581401.

[29] I. Abnane, A. Idri, I. Chlioui, and A. Abran, "Evaluating ensemble imputation in software effort estimation," *Empirical Software Engineering*, vol. 28, no. 2, Mar. 2023, Art. no. 56, https://doi.org/10.1007/s10664-022-10260-0.

[30] H. Hairani, T. Widiyaningtyas, D. Prasetya, and A. Aminuddin, "Addressing Imbalance in Health Datasets: A New Method NR-Clustering SMOTE and Distance Metric Modification," *Computers, Materials & Continua*, vol. 82, no. 2, pp. 2931–2949, Feb. 2025, https://doi.org/10.32604/cmc.2024.060837.

[31] P. Purnawansyah, A. Adnan, H. Darwis, A. P. Wibawa, T. Widyaningtyas, and H. Haviluddin, "Ensemble semi-supervised learning in facial expression recognition," *International Journal of Advances in Intelligent Informatics*, vol. 11, no. 1, pp. 1–24, Feb. 2025, https://doi.org/10.26555/ijain.v11i1.1880.

[32] H. Hairani, T. Widiyaningtyas, and D. D. Prasetya, "Feature Selection and Hybrid Sampling with Machine Learning Methods for Health Data Classification," *Revue d'Intelligence Artificielle*, vol. 38, no. 4, pp. 1255–1261, Aug. 2024, https://doi.org/10.18280/ria.380419.

[33] T. Widiyaningtyas, H. Hairani, D. D. Prasetya, U. Pujianto, and W. Caesarendra, "A Modified SMOTE with Noise Filtering and Manhattan Distance Metric Approach to Address Imbalanced Health Datasets,"

Engineering, Technology & Applied Science Research, vol. 15, no. 4, pp. 25452–25459, Aug. 2025, https://doi.org/10.48084/etasr.11925.

[34] D. E. Cahyani, A. D. Hariadi, F. F. Setyawan, L. Gumilar, and S. Setumin, "Classification of pediatric pneumonia using ensemble transfer learning convolutional neural network," *Bulletin of Electrical Engineering and Informatics*, vol. 13, no. 5, pp. 3411–3417, Oct. 2024, https://doi.org/10.11591/eei.v13i5.7825.

[35] S. S. Ali, J. Ren, K. Zhang, J. Wu, and C. Liu, "Heterogeneous Ensemble Model to Optimize Software Effort Estimation Accuracy," *IEEE Access*, vol. 11, pp. 27759–27792, 2023, https://doi.org/10.1109/ACCESS.2023.3256533.

[36] N. D. Ariyanta, A. N. Handayani, J. T. Ardiansah, and K. Arai, "Ensemble learning approaches for predicting heart failure outcomes: A comparative analysis of feedforward neural networks, random forest, and XGBoost," *Applied Engineering and Technology*, vol. 3, no. 3, pp. 173–184, Dec. 2024, https://doi.org/10.31763/aet.v3i3.1750.

[37] S. Lundberg and S.-I. Lee, "A Unified Approach to Interpreting Model Predictions," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, Long Beach, CA, USA, 2017, pp. 4766–4777, https://doi.org/10.48550/arXiv.1705.07874.

[38] M. T. Ribeiro, S. Singh, and C. Guestrin, "'Why Should I Trust You?': Explaining the Predictions of Any Classifier," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York, NY, USA, 2016, pp. 1135–1144, https://doi.org/10.1145/2939672.2939778.

[39] Z. Fatima *et al.*, "Explainable AI for IOT Devices and Robotic Communication Phishing Detection: A Machine Learning Approach Using LIME and SHAP," *Engineering, Technology & Applied Science Research*, vol. 15, no. 5, pp. 26478–26486, Oct. 2025, https://doi.org/10.48084/etasr.11595.

[40] F. Wilcoxon, "Individual Comparisons by Ranking Methods," *Biometrics Bulletin*, vol. 1, no. 6, pp. 80–83, Dec. 1945, https://doi.org/10.2307/3001968.

[41] S. Holm, "A Simple Sequentially Rejective Multiple Test Procedure," *Scandinavian Journal of Statistics*, vol. 6, pp. 65–70, Jan. 1979.

[42] N. Cliff, "Dominance statistics: Ordinal analyses to answer ordinal questions," *Psychological Bulletin*, vol. 114, no. 3, pp. 494–509, Nov. 1993, https://doi.org/10.1037/0033-2909.114.3.494.