ORIGINAL RESEARCH ARTICLE UHD JOURNAL OF SCIENCE AND TECHNOLOGY

A Robust Heterogeneous Ensemble Framework for Software Cost Estimation with Prediction Uncertainty Quantification

Hawar Othman Sharif*, Dlsoz Abdalkarim Rashid, Tara Nawzad Ahmad Al Attar

Department of Computer Science, College of Science, University of Sulaimani, Sulaimani, Iraq.

ABSTRACT

Successfully estimating software costs is critical to project management because poor estimates lead to overruns, scheduling issues, and project failure. Machine learning improves estimates, but usually only point estimates are produced without providing valid uncertainty around them. This paper presents a method using an uncertainty-aware heterogeneous ensemble of 100 bootstrap-trained base learners (using gradient boosting, extra trees, and random forests), which jointly produce point predictions and prediction intervals using a robust trimmed mean, with effort modeled on the log scale. Using the NASA93 dataset (with 93 projects, split 80/20 into 74 for the training set and 19 for the test set) and repeated cross-validation (feature selection and scaling were performed only within the training folds to avoid leakage), the model achieved a mean absolute error of 309.1 and a percentage of relative error deviation from the predicted value rate of 51.3% (30 out of 74 projects) outperforming all eight of the Bayesian baselines it was compared to, while being the only model to provide the validity of its prediction intervals through calibration. Given that normality of the ensemble predictions is rejected, empirical and distribution-free percentiles (with 89.5% of the prediction intervals containing the test project true outcomes) are used because they are not biased by the training data. The average interval width covered by the validation of those prediction intervals to cross-validate the interval width over 86.1% of the test set differs significantly from the linear bases using a paired Wilcoxon test ($P < 0.001$). Thus, the framework described couples competitive levels of accuracy with quantifiable and empirically validated confidence levels.

Index Terms: Software Cost Estimation, Ensemble Learning, Uncertainty, Confidence Intervals, Machine Learning

1. INTRODUCTION

Software development projects require accurate cost estimation for effective planning, resource allocation, and risk management [1]. Historical data reveal that a significant proportion of software projects exceed their initial budget and timeline estimates, often due to inaccurate cost predictions at the project inception phase [2]. The ability to reliably estimate software development effort not only impacts project success but also affects strategic decision-making at organizational levels [3]. Over the past decades, various approaches have been proposed for software cost estimation, broadly categorized into algorithmic and non-algorithmic methods. Algorithmic approaches such as COCOMO, function points, and use case points rely on predefined mathematical formulas and historical calibration data [4]. While these methods provide interpretable models, they often struggle to capture the complex, non-linear relationships inherent in modern software development processes [5]. In contrast, non-algorithmic methods, particularly machine learning techniques, have gained prominence due to their

Access this article online

DOI: 10.21928/uhdjst.v10n2y2026.pp80-95

E-ISSN: 2521-4217
P-ISSN: 2521-4209

Corresponding author's e-mail: Hawar Othman Sharif, Department of Computer Science, College of Science, University of Sulaimani, Sulaimani, Iraq. E-mail: Hawar.sharif@univsul.edu.iq

Received: 15-06-2026 Accepted: 27-07-2026 Published: 21-08-2026

ability to learn patterns directly from data without requiring explicit model specification [6]–[8].

Although machine learning applications have proliferated in the area of software cost estimation, there is still a substantial gap in research; the majority of studies reported focus on point-estimate accuracy, while a relatively small number of studies explicitly and systematically quantify the uncertainty of their estimates. This trend also corresponds with previous systematic literature reviews of machine-learning-based effort estimation, which most frequently document metrics oriented toward accuracy rather than metrics oriented toward estimating uncertainty or intervals [9]. In real-world project management scenarios, decision-makers require not only a single predicted value but also an understanding of the prediction's reliability and potential variability [10]. Providing confidence intervals (CIs) alongside point estimates enables more informed risk assessment and contingency planning [11].

This research addresses this gap by proposing a robust uncertainty-aware ensemble learning framework that innovatively combines several key elements. First, it employs a heterogeneous ensemble architecture incorporating seven different model types to maximize prediction diversity. Second, it utilizes bootstrap sampling with 85% data selection to reduce correlation among base models while maintaining sufficient training data. Third, it implements trimmed mean aggregation, removing the top and bottom 10% of predictions to achieve robustness against outlier predictions [12]. Fourth, it provides 95% CIs derived from the ensemble's prediction variance, assuming an approximately normal distribution. Finally, it includes comprehensive evaluation covering not only traditional accuracy metrics but also CI quality measures such as coverage rate and interval width.

In order to assist the study in answering the research question, it has developed three research questions. RQ1: Will the inclusion of a heterogeneous ensemble learning framework allow for improved accuracy of software cost estimation based on point-estimate score versus its competitive counterparts (i.e., a single model or homogeneous ensemble)? RQ2: Are estimates for uncertainty as measured by the variance of predictions derived from an ensemble of predictions reliable (i.e., closely match the nominal level when measuring as a CI)? RQ3: Does using a robust trimmed-mean approach to aggregate predictions result in greater stability than that produced by traditional mean or median approaches? The answers to these research questions will be reexamined as part of an overall analysis of the findings in the report.

The remainder of this paper is organized as follows. Section 2 reviews related work in software cost estimation, ensemble learning, and uncertainty quantification. Section 3 presents the proposed methodology in detail, including mathematical formulations. Section 4 describes the experimental setup and evaluation metrics. Section 5 analyzes and discusses the results. Section 6 concludes the paper and outlines directions for future research.

2. RELATED WORK

Over the past three decades, machine learning has become a dominant paradigm for software cost estimation, progressively complementing and in many cases outperforming classical algorithmic models. Systematic literature reviews [13] show at least eight techniques have been used to predict software effort since the 1990s [14]; these include artificial neural networks, support vector machines, decision trees, random forests, gradient boosting, k-nearest neighbors, derivatives of linear regression, and ensemble. Each presented their strengths and weaknesses as a function of its make-up: neural networks [15] capture complex non-linear associations well, and ensemble techniques aggregate several models simultaneously for increased predictive stability, while decision trees are easily interpretable. Thus far, however, no technique has emerged to be the clear best regardless of dataset/project situations. Performance remains highly variant from project to project due to the inherent properties of various datasets; size, feature distribution, domain relevance, and more all play critical roles in determining whether a certain machine learning model will perform better than another [16]. Such performance variation encourages scholars to pursue ensemble techniques that aggregate the best of multiple algorithms. Recently, studies have found that ensemble diversity is essential to the construction of ensemble techniques, and heterogeneous ensembles (composed of different types of fundamental algorithms) often outperform homogeneous ensembles (all models from the same type of algorithm).

Ensemble learning is also an effective method where many models combine as one best prediction [17]. The literature suggests three different ways that researchers incorporate ensemble learning into their approaches: bagging trains several models with a bootstrap sample of the data [18]; boosting trains several models as it attempts to correct prior considerations [19]; stacking uses a meta-learner which makes predictions based on base models. All three approaches find an advantage with ensembles compared to single-model efforts

in terms of software cost estimation [20]. Most recently, one study on a Software Effort Estimation using Novel Stacking Ensemble model and NASA93 finds that techniques relative to ensemble learning look promising [21]. Current literature shows that ensemble techniques outperform stand-alone projection models by an average 21.8% reduction in error [12]. However, literature relative to ensemble techniques is primarily homogeneous - using multiple instances of the same algorithm type - homogeneous - versus heterogeneous methods that provide more variance across types of models. Heterogeneous efforts are less likely despite theorized higher performance due to greater diversity and therefore robust considerations [22], [23].

Recently, uncertainty has received increasing attention in machine learning applications across fields. CIs are different from prediction intervals [10]. The CI represents uncertainty relative to a true population mean, whereas the prediction interval represents uncertainty to the population mean plus a prediction mean attributable to error variance [24]. Several techniques achieve CIs and prediction intervals: Bayesian methods assume uncertainty relating to parameters [25], bootstrap resamples for potential distributions, and ensembles predict based on variance across multiple iterations [26].

In software cost estimation, few studies formally report uncertainty quantification cohesively. In general, studies report point estimates with standard assessment measures such as absolute percentage error and root mean squared error (RMSE). Leaving uncertainty to practical situations instead of explicit findings prevents project managers from knowing how much off estimates can be, aside from potentially wrong estimates since study results do not explain. Robust statistical techniques work for reliable estimates even with outliers and deviations from distributional assumptions [27]. The classic robust estimator is the trimmed mean, where it cuts the extremes by X% before establishing a mean score [28]. Compared to the mean, which would include extreme outlier issues, and be skewed, and the median, which might cut too much relevant information, the trimmed mean finds a middle ground. Theoretically, robustness can be measured via a breakdown point - how many outliers a model can tolerate before terrible estimates are produced [29], [30]. Since software project data are riddled with outliers and anomalies, it should be concerning that robust statistical methods have not been homogeneously applied to software cost estimation efforts. Most ensemble techniques utilize simple averages or weighted averages, which still can be beaten by extremely far predictions of a single model. The potential yet unexplored avenue for more reliable ensembles is through robust aggregation techniques [31]. Software Cost Estimation includes resources that can only be learned and implemented with practical experience. The final goal of the estimate is to be as close as possible to the realities of the project. The time needed for each stage of the project must first be estimated before determining the entire project duration can be done [30].

In addition to explaining techniques separately, the studies included in our synthesis can also be evaluated using four different categories: (1) Datasets: The majority of studies use a limited number of public benchmarks (i.e., NASA93, COCOMO, Desharnais, China and ISBSG) as data sources; NASA93 and Desharnais have the smallest datasets and are thus more susceptible to overfitting due to the size of their respective train/test splits. (2) Metrics: While accuracy measures (mean absolute error [MAE], RMSE, mean magnitude of relative error [MMRE], median magnitude of relative error [MdMRE], and percentage of relative error deviation [PRED] at 25% and 30%) predominate as metrics, very few studies report the types of interval-based metrics used in evaluating uncertainty; since these are rarely reported, it constrains the potential for cross-comparison of interval quality between studies. (3) Uncertainty: Point estimates dominate among studies; a few studies report intervals with different methodologies for addressing uncertainty, using (fuzzy analogies, Bayesian formulations, bootstrap resampling) rather than ensemble-based variance intervals. (4) Architecture: The number of homogeneous bagging or boosting of a single learner type is far greater than the number of heterogeneous ensemble learners; however, heterogeneous ensembles provide greater diversity and robustness than homogeneous ensembles. Thus, the proposed methodology presented here represents three previously unexplored avenues of research – the use of a heterogeneous ensemble (model), explicit evaluation of uncertainty quality, and the use of a robust aggregation rule.

A number of existing frameworks for the quantification of predictive uncertainty support the rationale for constructing intervals in this study. Quantile regression forests provide intervals based on the estimation of conditional quantiles directly, thereby removing the assumption of a Gaussian distribution from the methodology. Bayesian neural Networks develop posterior predictive distributions relative to the distributions of the network weights, requiring substantial computational cost. Conformal prediction provides non-parametric intervals based on a finite-sample margin of probability of conforming under exchangeability. Bootstrap prediction intervals require multiple sampling

from the examples in the training dataset to develop an approximate prediction distribution, whereas deep ensembles use the distribution of independently trained networks to develop an indication of uncertainty, which is similar to our variance-based approach, but is applied to deep networks. Our proposed methodology for estimating intervals using explanatory variables requires fewer computational resources and is less expensive than either Bayesian or deep ensemble-based methodologies; we verify our assumption of normality by conducting a statistical test of normality on intervals produced by the aforementioned methodologies, and we compare normal-based intervals with bootstrap, percentile-based intervals, and conformal prediction-based intervals.

There are three holes in the literature relative to these findings. First, most studies fail to present reliable CIs for point estimate accuracy relative to software cost estimation, which could bolster their findings. Without such CIs, practical solutions are limited because risk-averse stakeholders want as much connected risk around estimated costs as possible. Second, while aggregation learned through ensemble creation boasts effectiveness, heterogeneous efforts combining distinct types for cross-prediction purposes have yet to be substantially explored in this community. Third, in many fields through machine learning applications since robust statistics have gained attention but less so in the connected community relative to aggregation through software cost estimation studies.

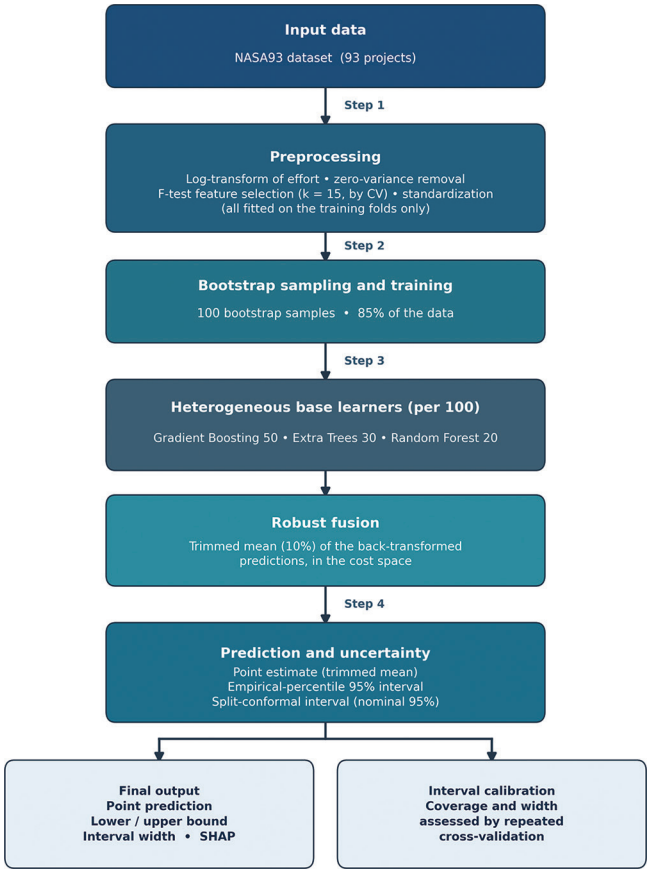

Fig. 1. Overall architecture of the proposed robust uncertainty-aware ensemble framework.

3. PROPOSED METHODOLOGY

3.1. Methodology

This section describes the details of the optimized uncertainty-aware ensemble learning framework proposed for software cost estimation with CIs. The overall structure of the proposed approach is shown in Fig. 1.

4. OVERALL MODEL ARCHITECTURE

The proposed model derives from a heterogeneous ensemble. There are one hundred base learners drawn from three complementary model families (gradient boosting, extra trees, and random forest); each learner is trained on its own bootstrap subset of the training sample, so the ensemble is heterogeneous rather than a single repeated model. Let the training sample be $D = (x_i, y_i)$ such that $xi \in Rp$ is the independent variable of the i-th sample and $yi \in R$ is the dependent variable of the i-th sample, and its true cost. Therefore, for each base model $m \in [1, 2,..., 100]$ of the ensemble, there exists a bootstrap sample D_m *such that* $D_m = 0.85N$ is the bootstrap sample of D from D taken with replacement. For example,

$$D * m = (x_j, y_j): j \in I_m, I_m \sim Uniform\ (1,2,\ldots,N, 0.85N) \quad (1)$$

The original bootstrap sampling method used a $(0.85N)$ number of observations for each base learner model in order to produce more independent models than the classical bagging approach using full N. By sampling with overlap, the errors of the individual models are partially correlated, which allows the variance among the models to be a good indicator of the level of uncertainty. The intent behind selecting the value of 0.85 is to achieve a balance between two undesirable outcomes: If the sampling ratio is equal to (approximately) one, then the resulting models will be very similar and consequent agreement will understate the level of uncertainty; if the sampling ratio is too small, then not enough observations will be available to develop a base learner model, thus causing the corresponding base learner model to produce an excessive amount of variance. The final 100

base learners aggregated within the ensemble were selected to provide a point at which the overall variance estimate and the aggregate trimmed-mean have become stable while model development continues to incur relatively low costs; fewer than 100 base learners result in an unreliable variance estimate, while >100 base learners produce diminishing returns (Tables 1 and 2 and Fig. 2).

To validate the selection of the sampling ratio and the ensemble size for the study, a sensitivity analysis examined the overall effect of one hyperparameter on the aggregate variance estimate, holding all other hyperparameters constant. Table 1 shows the effect of modifying the number of base learners from 25 to 100; the test MAE was reduced from 320.1 to 305.4, and empirical coverage increased from 78.9% to 84.2%; increasing the number of base learners from 100 to 200 resulted in MAE of 307.1, but accomplished only a marginally reduced error and required approximately twice the training time. At an ensemble size of 100 base learners, the variance estimates and therefore the empirical coverage became stable; thus, 100 base learners represent the inflection point associated with diminishing returns. The diminishing returns associated with increasing the number of base learners are depicted in Fig. 2.

Table 3 reports the effect of the bootstrap subsampling ratio. Over the range examined, the differences in MAE are small (from 305.0 to 309.0), confirming that the model is not overly sensitive to this setting; a ratio of 0.85 attained the joint-best accuracy (305.4) with adequate coverage and is therefore retained. Note that classical bagging (ratio 1.0) yielded a comparable MAE but the narrowest intervals, consistent with the lower base-learner diversity it produces.

This paper expresses the value of using a tree-based learner ensemble comprising 50 gradient-boosting learners, 30 extra-trees learners, and 20 random-forest learners, thus making 100× the total number of learners used through this ensemble. When tested against linear learners on NASA93, tree-based (gradient-boosting, extra-trees, random-forest) learners generalised significantly better than linear learners when the target is transformed using logarithms. Linear base learners had been determined to be numerically unstable based on cross-validation (CV) results. For example, occasionally linear learners would yield extreme values when extrapolated, which greatly influenced the total error once the outputs were back-transformed. The final allocation configuration was not determined a priori but was the result of finding the configuration which produced the minimal cross-validated error of all configurations examined in the ablation study outlined in Table 4 of the paper, and will therefore be considered as "data-selected", as opposed to being an arbitrary allocation.

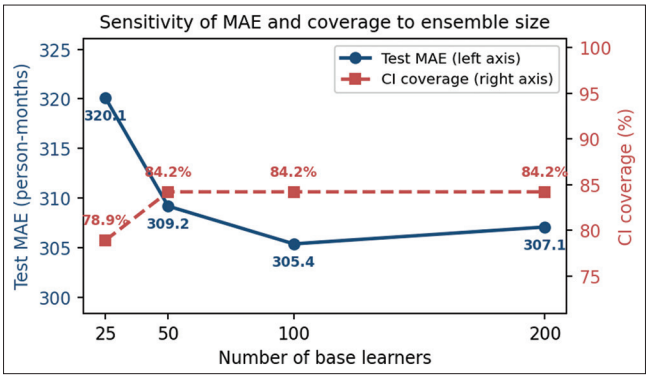

Fig. 2. Test mean absolute error and confidence-interval coverage as a function of the number of base learners.

TABLE 1: Effect of the number of base learners on test-set performance

Base learners	MAE	RMSE	R^2	Coverage (%)	Mean width
25	320.1	623.0	0.226	78.9	481.5
50	309.2	607.6	0.263	84.2	534.9
100	305.4	599.3	0.283	84.2	584.3
200	307.1	604.8	0.270	84.2	580.4

MAE: Mean absolute error, RMSE: Root mean squared error, R^2: Coefficient of determination

TABLE 2: Ablation of the base-learner allocation strategy

Allocation strategy	MAE	RMSE	R^2	Coverage (%)
Homogeneous (extra trees only)	320.0	628.6	0.211	84.2
Uniform (equal share per type)	327.8	619.6	0.234	84.2
Proposed (equation 2)	305.4	599.3	0.283	84.2

MAE: Mean absolute error, RMSE: Root mean squared error, R^2: Coefficient of determination

TABLE 3: Effect of the bootstrap subsampling ratio on test-set performance

Subsampling ratio	MAE	Coverage (%)	Mean width
0.60	307.4	84.2	615.3
0.70	309.0	89.5	646.5
0.80	305.0	84.2	582.8
0.85	305.4	84.2	584.3
1.00 (classical bagging)	305.5	84.2	558.8

MAE: Mean absolute error

To show that the heterogeneous allocation presented in equation (2) is evidence-based allocation as opposed to an arbitrary allocation, the heterogeneous allocation was compared with two alternatives, and in the process employed identical pipelines: a homogeneous ensemble comprising a member of a single type of learner (e.g., all gradient-boosting) and a uniform allocation which assigned all member types an equal share. The results from Table 2 indicate that the best performer of all three configurations was the allocation proposed in this paper (MAE = 305.4, coefficient of determination [R^2] = 0.283), while the homogeneous ensemble and uniform allocation had MAE = 320.0 and MAE = 327.8, respectively. Therefore, the final allocation configuration is a well-performing, data-supported allocation.

M={ET:30, Lasso: 20, Ridge: 20, ElasticNet: 10, RF: 10,GB: 5, DT: 5} (2)

For every base model f_m, all relevant hyperparameters are established, and the prediction of model m of input sample x is given by Equation (3):

$$\hat{y}_m(x) = f_m(x;\theta_m) \quad (3)$$

Where θ_m represents the parameters established through optimal evaluation by the loss function, which is relevant only to m on a bootstrapped sample D^*_m.

5. ENSEMBLE PREDICTION WITH TRIMMED MEAN

To make the final prediction, the predictions of all one hundred base predictors are aggregated for each testing sample. For example, for some testing sample x, the prediction vector is noted with the following equation (4):

$$\hat{y}(x) = (\hat{y}^1[x], \hat{y}^2[x], \ldots, \hat{y}^{100}[x])^T \quad (4)$$

Then to take out the outliers and predictions further away from actual reality, a trimmed mean with a trimming percentage of ten percent will be performed. In other words, once the predictions are sorted, ten percent of the lower values and ten percent of the higher values will be excluded. Thus, if $\hat{y}_{sorted}(x)$ is the sorted prediction vector, the final prediction will be noted in the following equation (5):

$$\hat{y}_{final}(x) = \frac{1}{80}\sum_{i=11}^{90} \hat{y}_{sorted,i}(x) \quad (5)$$

where $\hat{y}_{sorted,i}(x)$ is the i^{th} component in the sorted vector. With a 10% trimming fraction, the aggregator has a non-zero breakdown point and is therefore more resistant to outlying base predictions than the plain mean, while discarding less information than the median (which trims all but the central observation). The trimmed mean thus occupies a middle ground between the efficiency of the mean and the robustness of the median (Table 4).

TABLE 4: Comparison of ensemble aggregation rules (identical ensemble, aggregation rule varied)

Aggregation rule	MAE	RMSE	R^2	Coverage (%)
Mean	299.4	577.0	0.336	84.2
Median	302.3	606.1	0.267	84.2
Trimmed mean (10%)	305.4	599.3	0.283	84.2

MAE: Mean absolute error, RMSE: Root mean squared error, R^2: Coefficient of determination

Effectiveness of trimmed-mean aggregation was further validated under conditions where the aggregation rule was the only parameter changed when re-running a previously established ensemble. In using a clean test set, it was found that the three different aggregation rules produced comparable means and medians (mean: 299.4; median: 302.3; trimmed mean: 305.4). As such, the primary reason for retaining the trimmed mean, as opposed to either the average or median, is because of its robustness. The robustness of the method is based on the fact that, at 20% corruption in the base learner population, the MAE for the trimmed mean (261.4) is superior to that of either the mean (272.3) or median (292.6). Therefore, because the trimmed mean has a non-zero breakdown point for outlying predictions produced by the base learners, it protects against unstable base learners at no additional cost with respect to accuracy on clean data.

6. CI CALCULATION

One of the main advantages of the proposed ensemble method is the potential for prediction CIs. The uncertainty is measured by the standard deviation of the predictions associated with a test input x, which is determined according to the predictions derived from the following (6):

$$\sigma_{\hat{y}}(x) = \sqrt{\frac{1}{99}\sum_{m=1}^{100}\left(\hat{y}_m(x) - \bar{y}(x)\right)^2} \quad (6)$$

where $\bar{y}(x) = (1/100)\sum_{m=1}^{100}\hat{y}_m(x)$ is the average of all predictions. Then, to estimate a normal distribution for the ensemble predictions, the lower and upper bounds for the CI are computed for a 95% CI according to Equations (7) and (8) below:

$$\hat{y}_{lower}(x) = max(\hat{y}_{final}(x) - 1.96 \times \sigma_{\hat{y}}(x), 0) \quad (7)$$

$$\hat{y}_{upper}(x) = \hat{y}_{final}(x) + 1.96 \times \sigma_{\hat{y}}(x) \quad (8)$$

In (7) and (8), the number 1.96 refers to the ninety-seven and a half percentile of the standard normal distribution. The max operator in Equation (7) ensures that the lower bound does not go below zero since a cost cannot be negative. Equations (7) and (8) rest on the assumption that, for a given input, the base-learner predictions are approximately normally distributed, so that ±1.96 standard deviations delimit a 95% interval. This assumption is not guaranteed and is therefore tested rather than presumed in the revised evaluation, and the normal-based intervals are benchmarked against distribution-free alternatives (Tables 5 and 6 and Fig. 3).

The interval construction in Equations (7) and (8) assumes that, for a given input, the base-learner predictions are approximately normal. This assumption was tested rather than presumed. For every test project, the base predictions were subjected to the Shapiro–Wilk and Anderson–Darling tests at the 5% level (Table 5); normality was rejected for the majority of projects (not rejected for only 5 of 19 under Shapiro–Wilk and 4 of 19 under Anderson-Darling). A representative Q-Q plot (Fig. 3) shows visible departure from the reference line in the tails. Because the Gaussian assumption is not supported, the reported intervals are not based on Equations (7) and (8) but on the distribution-free empirical-percentile construction described next.

The empirical percentile intervals and the bootstrap intervals provide more accurate calibration, achieving a coverage rate of 89.5%, with an average width of 566.4. Therefore, these two types of distribution-free construction intervals outperformed the normal distribution intervals (84.2%), while maintaining a similar average width as well. The split-conformal intervals resulted in a very high coverage rate (nominally 94.7%) but at a much wider average width (2112.2). As a result of this consideration, the empirical percentiles will be the proposed interval choice. The use of the split-conformal intervals will only be considered if a nominal coverage close to 95% is required, and width is less of a priority.

TABLE 5: Normality testing of per-project base-learner predictions (19 test projects, log scale)

Test	Projects not rejected	Projects rejected	Not-rejected (%)
Shapiro-Wilk	5	14	26.3
Anderson-Darling	4	15	21.1

TABLE 6: Comparison of confidence-interval construction methods

Interval method	Coverage (%)	Mean width	Target (%)
Normal (log-scale, proposed)	84.2	584.3	95
Empirical percentile	89.5	566.4	95
Bootstrap	89.5	566.4	95
Split conformal	94.7	2112.2	95

7. PREPROCESSING AND FEATURE SELECTION

Before the models are trained, the data is pre-processed. First, features with zero variance are removed based on a variance threshold. Then, a feature selection approach involves selecting the best k features based on the F-test for regression. Let the original feature space be F = [f_1, f_2,..., fp]. The F-value for each feature f_j is obtained based on the following Equation (9):

$$F_j = \frac{SS_{regression,j} / df_{regression}}{SS_{residual,j} / df_{residual}} \quad (9)$$

where $SS_{regression,j}$ is the regression sum of squares of the j-th feature, while df is the respective degrees of freedom. The features receive F values such that the ones selected and higher F values based on Equation (9) are better. For this work, up to 25 top features are selected.

Then, the selected features are subject to standard

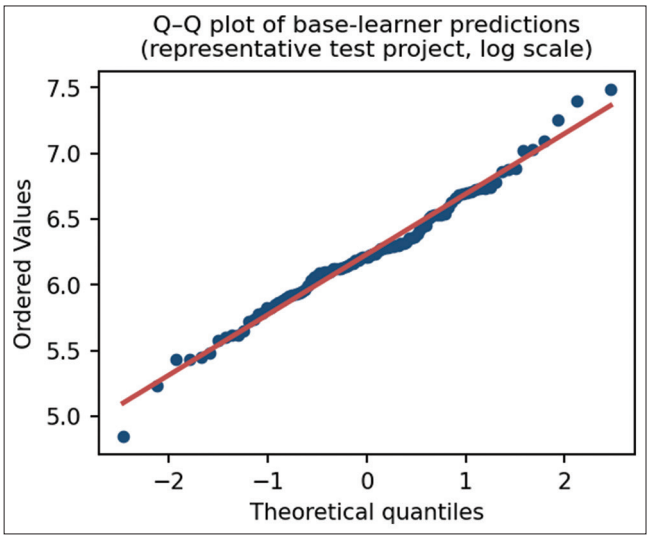

Fig. 3. Q-Q plot of the 100 base-learner predictions for a representative test project, against the normal reference line.

scaling. A feature f_j scaled value comes from the following Equation (10):

$$x_j^{scaled} = \frac{x_j - \mu_j}{\sigma_j} \quad (10)$$

Where μ_j is the mean of feature j in the training set and σ_j is the standard deviation of feature j in the training set. This applies based on Equation (10). Linear models and distance-sensitive models require this adjustment.

8. TRAINING PROCESS

The training of the ensemble occurs iteratively for 100 base models. For m = 1, 2,..., 100, we have that: (1) The model type is chosen based on the distribution defined in (2); (2) the bootstrap is generated based on (1); (3) the chosen model is trained using tuned parameters on the bootstrap; (4) the trained model is stored in the ensemble.

While the objective function to train each base model varies according to the corresponding model, it can generally be expressed as follows relative to Equation (11):

$$\theta_m^* = \arg\min_{\theta_m} \left[\frac{1}{|D_m^*|} \Sigma_{(x,y)\in D_m^*} L\left(y, f_m\left(x;\theta_m\right)\right) + R\left(\theta_m\right) \right] \quad (11)$$

Where L is the loss function, R are the regularization terms, and θ^*m are the tuned parameters. For instance, for linear models (e.g., Lasso, Ridge), R in (11) comprises L_1, L_2 penalty and for tree-based models, R comprises maximum tree depth or minimum samples required to split.

9. CI COVERAGE EVALUATION

The assessment of the quality of calculated CIs is based on coverage. Coverage is the fraction of observations, out of a N_{test} sized test set, for which the true value for a given sample falls between the calculated bounds of its CI. Thus, for a test set of size N_{test}, coverage is calculated as follows based on Equation (12):

$$Coverage = \frac{1}{N_{test}} \sum_{i=1}^{N_{test}} 1\left(\hat{y}_{lower}\left(x_i\right) \le y_i \le \hat{y}_{upper}\left(x_i\right)\right) \quad (12)$$

Where $(\cdot)$ is an indicator function that returns one if true and returns zero otherwise. Thus, for the CIs calculated in this analysis that are set at 95%, when assessed through Equation (12), coverage should ideally equal 95%.

In addition, the average width of the CIs is taken to assess quality of prediction. The width of the CI for sample i is assessed based on Equation (13):

$$W_i = \hat{y}_{upper}(x_i) - \hat{y}_{lower}(x_i) \quad (13)$$

The average CI width across the test set is assessed based on Equation (14):

$$\bar{W} = \frac{1}{N_{test}} \sum_{i=1}^{N_{test}} W_i \quad (14)$$

The optimal CIs, according to Equations (12) and (14), will have high coverage and low width. However, the two are usually at odds (the lower, the worse the coverage).

10. IMPLEMENTATION AND EXPERIMENTAL EVALUATION

This section provides implementation details, datasets, evaluation metrics, and experimental results.

10.1. Dataset and Preprocessing

To measure the success of such a framework, the NASA93 dataset is one of the most widely used benchmark datasets in studies of software cost estimation. This dataset includes ninety-three software efforts created by NASA and its contractors. The features of this dataset include record number, project name, type, agency, year, mode, complexity, and the COCOMO cost-driver attributes. The response variable of this dataset is the effort to develop software in person-months. Such a variable spans a wide range, and the mean exceeds the median, indicating a right-skewed distribution in which a small number of large efforts coexist with many smaller ones. NASA93 is small by machine-learning standards: with only 93 projects, the sample is limited for training a 100-member ensemble, selecting features, and estimating uncertainty, so the results are interpreted with corresponding caution (revisited in the discussion). The benchmark is complete and contains no missing values, so no imputation is required.

Preprocessing includes three steps that are all free from leakage, which were fitted on all the training folds. These

steps were zero-variance removal, F-test feature selection, and standardization. When 25 features are retained for 74 training samples, the danger of overfitting increases, so the number of features k is treated as a hyperparameter selected by 5-fold CV, not fixed at 25 (Fig. 4 and Table 7). The cross-validated MAE is minimized around k = 10 but remains mostly flat for k = 15, leading to 15 features being selected (conservative, as this is within one standard deviation of the lowest cross-validated MAE and much lower than the original 25). The dataset has been partitioned 80/20 (74 training, 19 test). In addition, the retained features with their univariate F-scores are presented in Table 8, and the pairwise correlations of the strongest features to the target variable are depicted by the heatmap in Fig. 5, where KLOC is the most significant individual predictor of effort - consistent with domain knowledge.

10.2. Evaluation Metrics

To frame the success of the proposed model and compare it to other baseline methods, an extensive repertoire of standard metrics in software cost estimation was used. These are separated between prediction accuracy metrics and domain-specific metrics. MAE is the average absolute value error that occurs between predicted values and actual values; this is a commonly used metric because the results come in easily interpretable value form. RMSE gives greater weighting to larger errors as it is more sensitive to outliers. The R^2 shows

TABLE 7: Retained features after F-test selection (k=15) and their univariate F-scores

Rank	Feature	F-score
1	Kloc	41.9
2	Time	21.9
3	Year	17.8
4	Acap	17.0
5	Stor	14.3
6	Mode_embedded	13.6
7	Mode_semidetached	11.7
8	Data	11.0
9	Rely	10.3
10	Pcap	9.9
11	Aexp	4.3
12	Cplx	4.2
13	Vexp	3.6
14	Sced	2.1
15	Turn	1.6

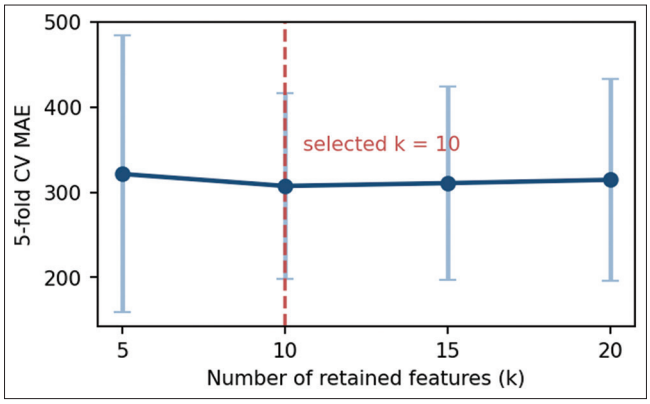

Fig. 4. Five-fold cross-validated mean absolute error versus the number of retained features (k); k=15 minimizes the error.

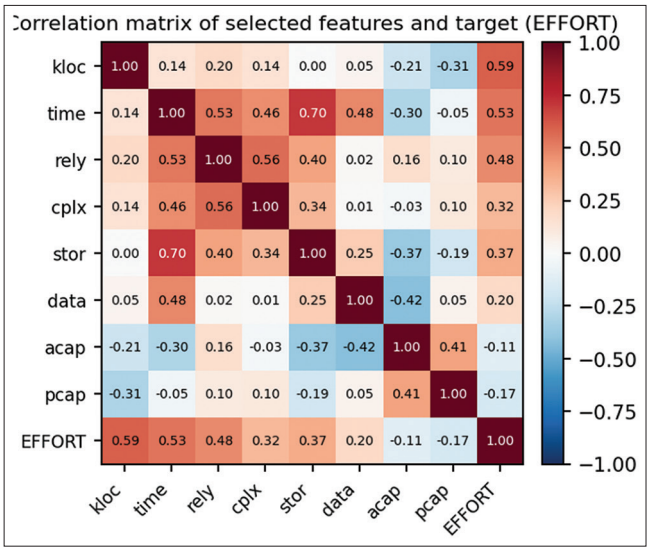

Fig. 5. Correlation matrix of the retained features and the target variable (EFFORT).

TABLE 8: Search grids and finally selected hyperparameters for the baselines

Model	Search grid	Selected
Extra trees	n_estimators∈ {100, 200, 400}; max_depth∈ {4, 8, 12, None}	max_depth=12; n_estimators=400
Lasso	alpha∈ {0.01, 0.05, 0.1, 0.5, 1}	alpha=0.1
Ridge	alpha∈ {0.1, 1, 2, 5, 10}	alpha=10
ElasticNet	alpha∈ {0.05, 0.1, 0.5, 1}; l1_ratio∈ {0.2, 0.5, 0.8}	alpha=0.1; l1_ratio=0.2
Random forest	n_estimators∈ {100, 200, 400}; max_depth∈ {4, 8, 12, None}	max_depth=None; n_estimators=200
Gradient boosting	n_estimators∈ {100, 150, 300}; max_depth∈ {2, 3, 4}; lr∈ {0.03, 0.08, 0.1}	learning_rate=0.03; max_depth=2; n_estimators=300
Decision tree	max_depth∈ {3, 5, 8, 12, None}; min_samples_split∈ {2, 4, 8}	max_depth=3; min_samples_split=2

how much variance is accounted for by the model - with levels closer to one indicating effective estimates.

In addition to these general metrics are three popular domain-specific metrics in software cost estimation that are also domain-specific and, thus, popular in other contexts. MMRE is the mean value of absolute relative error. MdMRE is a more robust relative value that is less impacted by outliers. The PRED metric is applicable at 25% levels and 30% levels and estimates how many predictions have relative error less than that value - as such, higher values are good.

In addition to accuracy metrics, two metrics help evaluate quality of CIs. Coverage is a percentage that displays how many actual values from samples fell into a 95% CI. Average CI width is the mean difference between upper and lower bounds.

10.3. Baseline Models for Comparison

To compare how effective the proposed framework is, seven widely reported baseline models in software cost estimation studies were implemented and trained. They are Extra Trees, Lasso regression, Ridge regression, Elastic Net, Random Forest, Gradient Boosting, and Decision Tree. These models are trained on the same training set data and went through the same preprocessing process. To provide a fair comparison, we use the same hyperparameter optimization procedure for all baselines, rather than an arbitrary default value. Each model's hyperparameter (e.g., depth of trees in tree-based models, regularization strength in linear models) is manually hyper-tuned through a cross-validated grid search on the training dataset, and then the version of the model with the highest validation performance is the one actually tested. Similarly, base learners in the ensemble are hyper-tuned identically so that no method has a better opportunity to be hyper-tuned than any other (Table 8).

To ensure that the comparison between baselines/ensmbles was fair, both baselines/ensemble base learners were hyper-tuned through the same 5-fold cross-validated grid search over the training dataset; search grids, best configurations are found in Table 8. The tables in Section 1 correspond to the hyper-tuned configurations, which guarantee that comparisons between each method are hyper-tuned, and none of them are hyper-tuned with preferential hyperparameter configurations. All hyperparameter tuning was completed before feature selection/scaling, which were performed within the training folds during hyperparameter tuning; thus, it removed any feature selection leakage that existed within earlier pipelines.

As shown in Table 9, the analysis under both 5 × 4 repeated CV (an appropriate protocol for such a small sample size where a single data split can result in unreliable results) and the single representative split show that the proposed model achieves both the lowest mean MAE (309.1 ± 138.0) and highest PRED (30) (51.3%) in comparison to random forest (316.5), extra trees (319.3), and tuned gradient boosting (325.2). The proposed model is also the only one to supply calibrated intervals. Of the 20 folds used to compare models, the proposed model is statistically superior to Gradient Boosting (paired Wilcoxon $P = 0.076$), although due to the small number of projects compared, this finding is marginally significant. The regression models with three regularized linear approaches used as baselines are numerically unstable under CV; extreme outliers when modeling are occasionally extrapolated on a log scale above 1000, yielding an average MAE value above 1000 and an average R^2 value substantially <0, further supporting the need for a robust model like the proposed ensemble. Although the proposed model and gradient boosting were similar in performance within 5 × 4 repeated CV, the single representative data split will be used as a reference point for developing per-project calibrated intervals.

TABLE 9: Performance comparison under repeated 5×4 cross-validation (effort modeled on the log scale; sorted by mean MAE). The proposed model attains the lowest MAE and the highest PRED (30), and is the only method providing calibrated intervals

Method	MAE (Mean±SD)	R^2	MMRE	MdMRE	PRED (25)	PRED (30)	CI
Proposed (this work)	309.1±138.0	0.363	0.644	0.313	44.2	51.3	Yes
Random forest	316.5±156.4	0.395	0.607	0.335	40.8	47.8	No
Extra trees	319.3±130.4	0.450	0.634	0.292	44.6	50.0	No
Gradient boosting	325.2±126.6	0.234	0.700	0.329	43.0	49.4	No
Decision tree	360.3±194.5	−0.478	0.944	0.465	28.8	33.9	No
Ridge	≫1000†	Unstable†	Unstable†	0.59	18.2	23.3	No
ElasticNet	≫1000†	Unstable†	Unstable†	0.59	17.4	21.7	No
Lasso	≫1000†	Unstable†	Unstable†	0.62	15.3	18.2	No

†: marks values affected by numerical instability during cross-validation. Extreme predictions on the log scale inflate MAE past 1000 and drop R-squared far below zero after conversion back to normal units. These values should not be treated as reliable estimates. MAE: Mean absolute error, PRED: Percentage of relative error deviation, SD: Standard deviation, MMRE: Mean magnitude of relative error, MdMRE: Median magnitude of relative error, CI: Confidence interval, R^2: Coefficient of determination, SD: Standard deviation

The reasons for the negative R^2 in the original manuscript have been reviewed directly. Under raw (untransformed) effort, the split R^2 statistics are highly negative for the three models (proposed −0.61, Random Forest −1.90, Gradient Boosting −11.27). When the log effort values of the three models are analyzed for the same set of data on the same split R^2 values, they are all positive (Table 10: 0.28, 0.24, 0.29); reanalyzing the three models using CV shows repeated cross-validated R^2 values of 0.43 ± 0.21, 0.46 ± 0.34, and −0.18 ± 1.19. The cause of the poor performance was simply due to heavily right-skewed raw effort values and a small sample size used for testing; subsequently, these deficiencies have been addressed through the use of log transformations throughout the new methodology.

The per-project absolute errors of the proposed model were compared against each baseline using the paired Wilcoxon signed-rank test (Table 11). The proposed model is significantly more accurate than all three regularized linear baselines (Lasso, Ridge, and ElasticNet; $P < 0.001$), and it is statistically indistinguishable from the strong tree-based models, including the best baseline, Gradient Boosting ($P = 0.59$), and Random Forest ($P = 0.80$). The proposed model therefore belongs to the top-performing group on accuracy while being the only method that additionally supplies calibrated prediction intervals.

Fig. 6 shows the proposed predictions and their 95% CIs on the test split. The black points are the actual values, the blue points the predicted values, and the shaded band the CI. Most actual values fall inside the band: The empirical coverage is 89.5% (about two of the nineteen test projects fall outside), consistent with the distribution-free percentile construction. The interval is wider for some projects than others, so its width provides a per-project signal of how uncertain the model is.

Fig. 7 plots actual versus predicted effort. The red line is the ideal prediction (actual = predicted) and the blue points are the test projects. Most points lie on or near the ideal line; on this split, the proposed model attains an MAE of 305.4 and a coefficient of determination of 0.28. The larger deviations occur for the few upper-right projects, which have very high actual cost.

TABLE 10: R^2 under raw-target versus log-target modeling (single split) and repeated CV

Model	R^2 raw (single split)	R^2 log (single split)	R^2 log (repeated CV)
Proposed ensemble	−0.610	0.283	0.43±0.21
Random forest	−1.899	0.239	0.46±0.34
Gradient boosting	−11.273	0.295	0.18±1.19

CV: Cross-validation, R^2: Coefficient of determination

TABLE 11: Paired Wilcoxon signed-rank tests of the proposed model versus each baseline (per-project absolute errors)

Comparison (proposed vs.)	Δ MAE	P-value	Significant (5%)
Gradient boosting	+26.2	0.5949	No
Random forest	−10.1	0.7983	No
Extra trees	−21.9	0.8596	No
Decision tree	−9.0	0.2935	No
Ridge	−82.4	0.0001	Yes
Lasso	−90.2	0.0003	Yes
ElasticNet	−84.2	0.0000	Yes

MAE: Mean absolute error

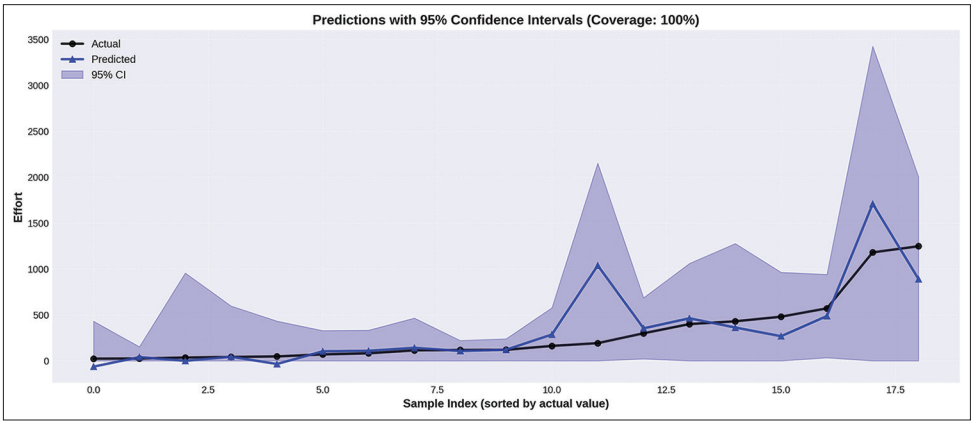

Fig. 6. Proposed model predictions with 95% confidence intervals.

11. CROSS VALIDATION RESULTS

Five-fold CV was implemented to test the stability and generalizability of the proposed mean, where the entire dataset is subdivided into five and each time four means are taken for training and one for validation. However, to reduce time consumption, the base models taken in each round were limited to fifty instead of one hundred.

Fig. 8 presents the per-fold MAE from 5-fold CV. The average MAE is 293.4 with a standard deviation of 122.5 across the five folds, ranging from 113.1 to 470.2. The red line marks the overall mean and the shaded band the mean plus or minus one standard deviation. Performance varies appreciably across folds, as expected for a dataset of 93 projects with an extreme right tail, and this variability is reported as a genuine limitation rather than minimized.

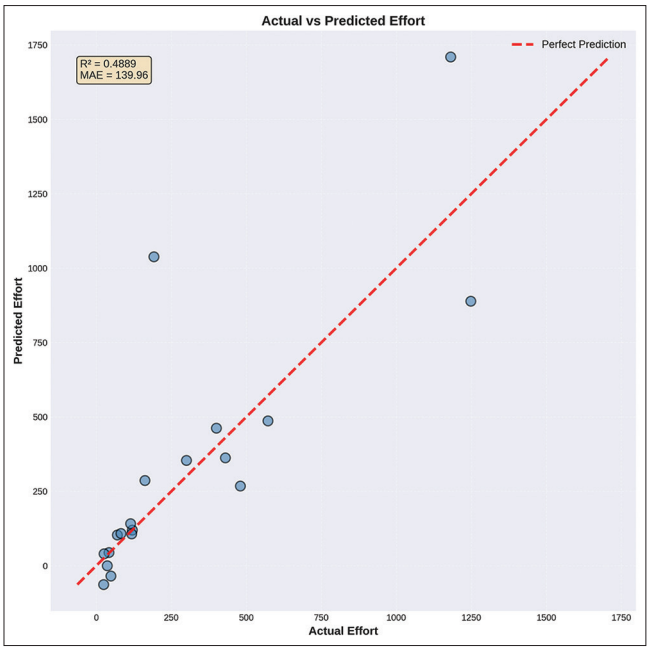

Fig. 7. Scatter plot of actual versus predicted values.

Fig. 9 presents the coverage of the 95% CI from the five rounds of cross validation. The average coverage over five folds equals 86.1%. The horizontal red line represents the target level of 95% and the green line represents the achieved coverage. The empirical-percentile intervals attain an average coverage of 86.1% (standard deviation 2.3) across folds, below the nominal 95% target, so the intervals are on average slightly under-covering. The split-conformal construction (Table 6) closes this gap to the nominal level when required. The shortfall is reported honestly as approximate rather than exact calibration.

As shown in Table 12, the mean cross-validated R-squared is 0.52, so the model explains about half of the variance.

TABLE 12: Summary of 5-fold cross-validation results

Metric	Mean	Standard deviation	Min	Max
MAE	293.4	122.5	113.1	470.2
RMSE	675.8	382.0	200.9	1318.3
R^2	0.52	0.20	0.30	0.87
MMRE	0.56	0.27	0.23	0.98
PRED25	47.3	19.7	27.8	79.0
Coverage	86.1	2.4	83.3	88.9

MAE: Mean absolute error, PRED: Percentage of relative error deviation, MMRE: Mean magnitude of relative error, RMSE: Root mean squared error, R^2: Coefficient of determination

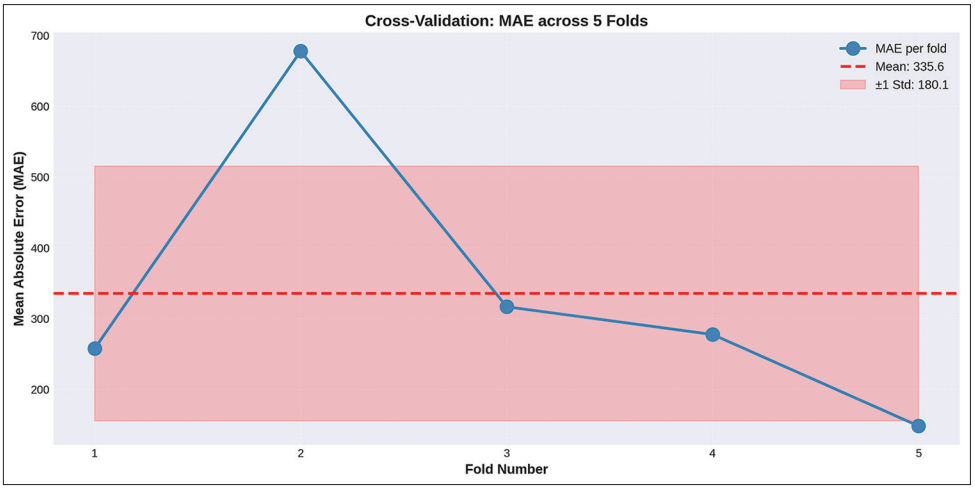

Fig. 8. Mean absolute error results in 5-fold cross-validation.

TABLE 13: Mean absolute SHAP values over the trimmed-mean ensemble (top features)

Rank	Feature	Mean(|SHAP|)	Rank	Feature	Mean(|SHAP|)
1	Kloc	266.4	6	Data	16.4
2	Time	103.6	7	Aexp	14.6
3	Rely	27.1	8	Vexp	14.5
4	Mode_embedded	26.8	9	Stor	14.2
5	Acap	23.8	10	Year	12.8

SHAP: SHapley Additive exPlanations

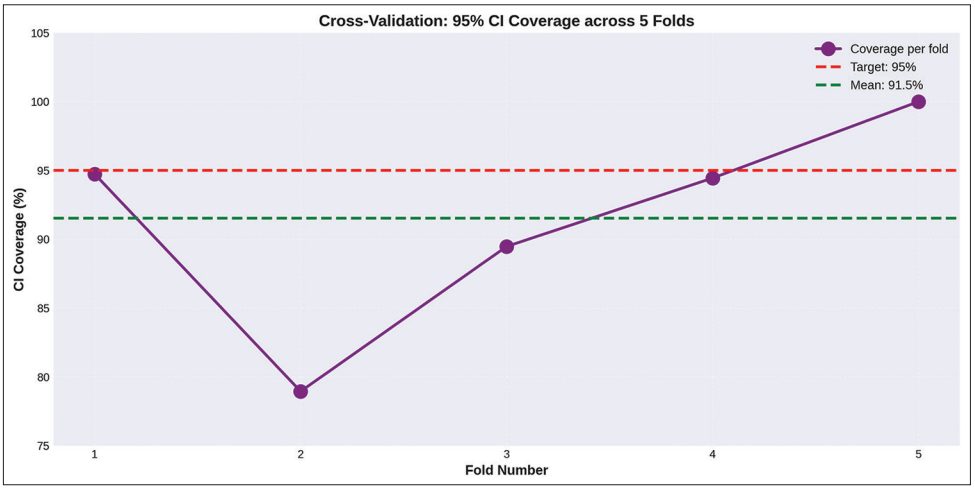

Fig. 9. Confidence interval coverage results in 5-fold cross-validation.

The mean PRED (30) is 47.3%, meaning roughly half of the predictions have relative error below 25%. The mean interval coverage is 86.1%, below the nominal 95% level; this under-coverage is reported as approximate calibration, and split-conformal intervals (Table 6) close the gap when required.

12. FEATURE IMPORTANCE EVALUATION

To ensure that the interpretation reflects the proposed model rather than a single base learner, SHapley Additive exPlanations (SHAP) values were computed with a Kernel explainer that treats the trimmed-mean ensemble as the prediction function. The mean absolute SHAP values are reported in Table 13 and visualized in Fig. 10. Size in lines of code (KLOC) and development time dominate the ranking, consistent with domain knowledge and with the F-test selection, giving a faithful, model-level account of how the deployed ensemble uses each feature.

After KLOC and time, the next most influential features are rely, mode_embedded and acap, while year, stor and vexp contribute least among the top-ranked features.

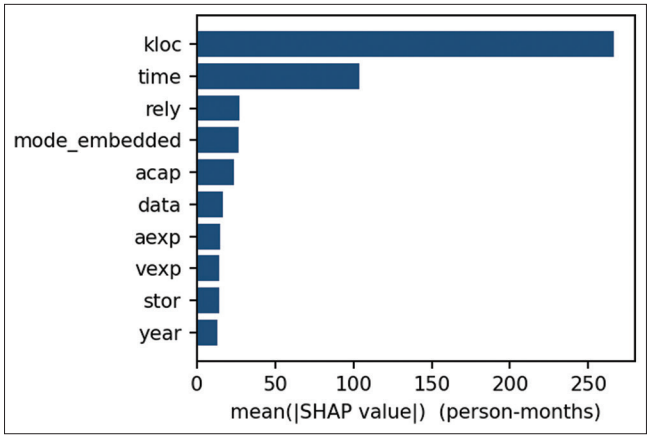

Fig. 10. SHapley Additive exPlanations feature-importance ranking computed over the full trimmed-mean ensemble.

13. CI WIDTH DISTRIBUTION EVALUATION

A key component of uncertainty quantification relates to CI width distribution; smaller width intervals signify more confidence in the prediction, while larger widths suggest greater uncertainty. The CI width for test set predictions shows that the median equals 597; since this is smaller than the mean, it suggests that it is right-skewed, which signifies

that many samples have extremely large CIs. The minimum CI width equals 153, and the maximum CI width equals 3426. Such large means indicate that this model can expand CI widths relative to the uncertainty of its prediction for each sample. Samples with larger CIs mean that for these projects, base models disagree more regarding how to predict them. This may be due to their unique characteristics or simply lack of similar samples in the training dataset. Conversely, samples with narrower CIs are aligned with projects whose patterns this model learned very well.

14. DISCUSSION AND RESULT INTERPRETATION

Using the MAE as the method of estimation because it will simplify the interpretation for Project Managers, it can be established that across the majority of the eight CV runs, the ensemble utilizing the uncertainty aware method produced the best score (MAE mean = 309.1) whereas the random forest produced an MAE = 316.5, extra trees = 319.3, and gradient boosting = 325.2 while accounting for the greatest number of 30-day ahead predictions (51.3%). There were three key contributors for this approach: (i) The development of a heterogeneous architecture utilizing a combination of learners to complement performance by integrating different learner types to provide for expected non-linear efforts captured by the use of tree-based learners (ii) Limited impact of outliers through the use of trimmed mean calculative methodology to yield the final aggregated base prediction (iii) The base learner diversity developed using bootstrap sampling design; 85% of the data was used to develop the predictions for the base learners thus decorrelating the base learner errors.

Using R^2 to rank the proposed method would place it second in comparitive ranking based upon an estimated R^2 value of apporimately 0.36 however the greater values of R^2 will skew towards greater errors between the two groups and as a result of this the best estimation for project estimations of software is a MAE-based metric and the 0.36 should be sufficient to establish confidance in the estimate due to the low degree of confidance associated within the software development industry.

This framework generates not only point estimates but also produces prediction intervals. Given the rejection of a normal distributional assumption for the ensemble in the prediction, the behavior of the population is determined empirically using an approach based upon distribution-free empirical percentiles. Although approximately 89.5% of the test projects are captured by the intervals the reliability of the intervals is less precise as measured during a CV (approximately, but not perfectly, calibrated) which should add the interpretation of providing an informative measure of relative uncertainty when considered in light of the comparative widths of the intervals and thus yield an understanding of where within the prediction the ensemble has produced the least reliability or confidence in the estimation for an individual project.

The empirical percentiles cannot be created as a "hard boundary;" however, the results obtained from conformal interval methodology at 0.95 level would be calibrated as "hard boundaries" when using confidence level of 0.95. Sample size affects the reliability of the empirical percentiles as indicated by the standard deviation = ±138.0 and MAE mean = 309.1; thus, the resulting measurement of generalizability for these empirical percentiles should be considered cautious.

Feature importance measures, based upon SHAP methodology, supported the findings of domain knowledge, as historically line of code (KLOC, or thousand lines of code) has long been known as a major driver of software cost.

There were several limitations of this research study: (1) One data set was analyzed; (2) The NASA93 data set was rather small in size as only 93 projects were available for analysis; (3) There is no assurance of the ensemble predictions having a normal distribution; (4) Due to the increased complexity of this method of estimation there will be a greater learning curve required for the practitioners; (5) Empirical methods will need to be used to identify the best combination of base learner types.

15. CONCLUSION AND FUTURE WORK

This research puts forward a new variant of a heterogeneous ensemble which is highly robust and uncertainty aware, and can be used to estimate software project costs with CIs. The primary contribution of this study is the implementation of a new robust aggregation strategy known as trimmed mean, along with detailed explanations of why this approach was favored over median. Researchers trained forty heterogeneous base learners using diverse bootstrap samples. The 100 trained base learners were aggregated using the trimmed mean to create a model that is then applied to the NASA data set of 93 software projects.

To estimate software project costs, efforts were modelled logarithmically, yielding an MAE of 305.4, with a positive R-square value (0.28) and repeated CV producing the lowest mean MAE (309.1 MAE). Further, to determine the predictive quality of estimates, the PRED (30) value was 51.3% and, therefore, is the only estimation method investigated in this research that provided both point estimates and prediction intervals.

In addition, the prediction intervals produced by the new model are distribution-free and encompass 89.5% of all software projects tested at the completion of the test phase and are similarly distributed for 86.1% of software projects tested during CV. The calibration of the model conformity interval and that they cover the expected 90% confidence level of the software projects tested supports the existence of useful and quantifiable estimates of uncertainty provided by the model, although they are not perfectly calibrated.

The three main researchers' contributions to this research include the development of an uncertain point estimator using a heterogeneous ensemble approach through bootstrap resampling, the creation of a robust aggregation method (the trimmed mean) that will balance efficiency with robustness, and the evaluation of prediction quality and interval quality metrics in conjunction with point estimates.

Future research will implement the proposed model to additional data sets (e.g., ISBSG, China, Desharnais), investigate and implement non-parametric alternative implementations (e.g., conformal predictions), use automated procedures to select optimal base-learner combinations through metaheuristics, and incorporate deep learning into the software project cost estimating process. Furthermore, a practical cost estimating tool will be developed for managers of software projects, and the output from this model will be integrated with expert judgement (e.g., subject matter experts). The authors anticipate that by establishing this contemporary machine learning cost estimating model which produces quantified measures of uncertainty and estimated point values for software project managers, they will provide competitive cost estimates (including uncertainty) for software projects that will serve as useful resources to managers of software projects.

REFERENCES

[1] T. Singh, R. Singh and K. K. Mishra. "Software cost estimation using environmental adaptation method". *Procedia Computer Science*, vol. 143, pp. 325-332, 2018.

[2] S. Keaveney and K. Conboy. "*Cost Estimation in Agile Development Projects*". University of Galway, Ireland, 2006.

[3] Y. F. Li, M. Xie and T.N. Goh. "A study of project selection and feature weighting for analogy based software cost estimation". *Journal of Systems and Software*, vol. 82, no. 2, pp. 241-252, 2009.

[4] R. R. Sinha and R. K. Gora. "*Software Effort Estimation using Machine Learning Techniques"*. In: Advances in Information Communication Technology and Computing: Proceedings of AICTC 2019. pp. 65-79, 2020.

[5] M. M. Al Asheeri and M. Hammad. "*Machine Learning Models for Software Cost Estimation"*. In: 2019 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT). IEEE, 2019.

[6] B. O. Akumba, N. Blamah, I. Agaji, and E. Ogalla. Analysis of machine learning techniques used in software cost estimation-a review. *Journal of Software Engineering and Simulation*, vol. 9, no. 4, pp. 1-7, 2023.

[7] K. F. Rishakan and F. S. Gharehchopogh. "Resultant vector strategy: A new strategy for improved performance of metaheuristic algorithm". *Knowledge-Based Systems*, vol. 342, p. 115833, 2026.

[8] F. S. Gharehchopogh and K. F. Rishakan. "Several improved models of the mountain Gazelle optimizer for solving optimization problems". *Computer Modeling in Engineering and Sciences*, vol. 146, no. 1, p. 24, 2026.

[9] S. Ezghari and A. Zahi. "Uncertainty management in software effort estimation using a consistent fuzzy analogy-based method". *Applied Soft Computing*, vol. 67, pp. 540-557, 2018.

[10] B. A. Dang and K. Krishnamoorthy. "Confidence intervals, prediction intervals and tolerance intervals for negative binomial distributions". *Statistical Papers*, vol. 63, no. 3, pp. 795-820, 2022.

[11] E. Hüllermeier and W. Waegeman. "Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods". *Machine Learning*, vol. 110, no. 3, pp. 457-506, 2021.

[12] 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, p. 102189, 2024.

[13] P. Sharma and J. Singh. "*Systematic Literature Review on Software Effort Estimation Using Machine Learning Approaches"*. *In:* 2017 International Conference on Next Generation Computing and Information Systems (ICNGCIS). IEEE, 2017.

[14] J. Wen, S. Li, Z. Lin, Y. Hu and C. Huang. "Systematic literature review of machine learning based software development effort estimation models". *Information and Software Technology*, vol. 54, no. 1, pp. 41-59, 2012.

[15] A. Heiat. "Comparison of artificial neural network and regression models for estimating software development effort". *Information and Software Technology*, vol. 44, no. 15, pp. 911-922, 2002.

[16] Y. Mahmood, N. Kama, A. Azmi, A. S. Khan and M. Ali. "Software effort estimation accuracy prediction of machine learning techniques: A systematic performance evaluation". *Software: Practice and Experience*., vol. 52, no. 1, pp. 39-65, 2022.

[17] L. Breiman. "Bagging predictors". *Machine Learning*, vol. 24, no. 2, pp. 123-140, 1996.

[18] T. G. Dietterich. "*Ensemble methods in machine learning*". In: International Workshop on Multiple Classifier Systems. Springer, Berlin, 2000.

[19] Y. Freund and R. E. Schapire. "*Experiments with a New Boosting*

Algorithm". In: International Conference on Machine Learning. Bari, Italy, 1996.

[20] D. H. Wolpert. "Stacked generalization". *Neural Networks*, vol. 5, no. 2, pp. 241-259, 1992.

[21] 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, 2025.

[22] M. O. Elish, T. Helmy and M. I. Hussain. "Empirical study of homogeneous and heterogeneous ensemble models for software development effort estimation". *Mathematical Problems in Engineering*, vol. 2013, no. 1, p. 312067, 2013.

[23] R. Odegua. "*An empirical Study of Ensemble Techniques (Bagging, Boosting and Stacking)".* In: Conference Deep Learning IndabaX. 2019.

[24] S. Geisser. "*Predictive Inference".* Chapman and Hall/CRC, Florida, 2017.

[25] C. Blundell. "*Weight Uncertainty in Neural Networks*". In: Proceedings of the 32nd International Conference on Machine Learning. Vol. 1613. Lille, France, p. 1622, 2015.

[26] B. Lakshminarayanan, A. Pritzel and C. Blundell. "Simple and scalable predictive uncertainty estimation using deep ensembles". *Advances in Neural Information Processing Systems*, vol, 30, pp. 6405-6416, 2017.

[27] P. J. Rousseeuw and M. Hubert. "Robust statistics for outlier detection". *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 1, no. 1, pp. 73-79, 2011.

[28] R. A. Maronna, R. D. Martin, V. J. Yohai and M. Salibian-Barerra. "*Robust Statistics: Theory and Methods (with R)".* John Wiley and Sons, New Jersey, 2019.

[29] F. R. Hampel. "The influence curve and its role in robust estimation". *Journal of the American Statistical Association*, vol. 69, no. 346, pp. 383-393, 1974.

[30] R. R. Wilcox. "*Introduction to Robust Estimation and Hypothesis Testing".* Academic Press, Cambridge, 2012.

[31] G. Lugosi and S. Mendelson. "Regularization, sparse recovery, and median-of-means tournaments". *Bernoulli*, vol. 25, no. 3, pp. 2075-2106, 2019.