International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18(8s) 2026 pp. 1105-1121
https://doi.org/10.70917/ijcisim-2026-3415
Article

EOELM: An Explainable Optimized Ensemble Learning Model for Intelligent Software Cost Estimation

Manas Prasad Rout[1], Sabyasachi Pattnaik[2], Banaja Basini Rath[3], Nabin Kumar Naik[4], Umashankar Ghugar[5]

[1,2,3]Department of Computer Science, Fakir Mohan University, Balasore, Odisha
manasprasadrout@gmail.com,
spattnaik1965@rediffmail.com
banaja.ratha@gmail.com
[4,5]Silicon Institute of Technology, Sambalpur, Odisha
nabin.naik@silicon.ac.in
ughugar@gmail.com

Abstract: Accurate early cost estimation of software is critical to effective project planning, budgeting, resource allocation and risk management. However, the traditional algorithmic models and most of the contemporary machine learning models are restricted in terms of prediction power, rely on manually set hyperparameters and do not provide model interpretability, which limits their practical application in software project management. In this research, to solve these problems for intelligent software cost estimate, an Explainable Optimized Ensemble learning Model (EOELM) is proposed. The proposed approach is a single estimate model which includes intelligent data preparation, hybrid feature selection, hyperparameter optimization based on Grey Wolf Optimizer (GWO), weighted stacking ensemble learning and Explainable Artificial Intelligence (XAI). Datasets of software projects are pre-processed using missing value imputation, outlier reduction, normalization and feature engineering. A hybrid feature selection method based on Correlation Analysis, Mutual Information, Recursive Feature Elimination and the Boruta algorithm is used to discover the most relevant software cost drivers. Then, the hyperparameters of four complimentary base learners, eXtreme Gradient Boosting (XGBoost), Light Gradient Boosting Machine (LightGBM), Random Forest (RF), and Artificial Neural Network (ANN), are fine-tuned by GWO, and their predictions are combined via a weighted stacking ensemble for improving the accuracy and robustness of the estimate. SHAP and LIME enhance the interpretability of the model and explain the software cost estimates at global and local level. The suggested system is validated on benchmark data sets as NASA93, COCOMO81, Desharnais, Maxwell and ISBSG using 10 fold cross validation. The performance evaluation is done using MAE, RMSE, MMRE, MdMRE, Pred(25) and R^2 and statistical significance of the findings is checked using Wilcoxon signed-rank and Friedman tests. The experimental findings show that EOELM is superior than the current software cost estimating approaches in prediction accuracy, generalization ability and interpretability. The framework presented is a dependable and transparent decision support system for the intelligent planning and cost estimation of software projects.

Keywords: Software Cost Estimation, Ensemble Learning, Explainable Artificial Intelligence, Grey Wolf Optimizer, XGBoost, LightGBM, Random Forest, Artificial Neural Network, SHAP, LIME, Early Software Project Estimation.

1. INTRODUCTION

Software cost estimation (SCE) is an essential discipline in software engineering as it influences project planning, budgeting, resource allocation, scheduling and risk management. Providing accurate estimates helps organizations to maximize the use of the resources, enhance planning of a project, decrease financial risks and boost chances of successful software delivery. On the other side, bad estimating typically leads to cost overruns, schedule

delays, low quality software, customer dissatisfaction, and project failure. Cost estimation of software is still an open research topic after many decades of study, due to increasing complexity of software systems, dynamic development environments, changing customer requirements and uncertainty in the early stages of software development [1]. Traditionally, software cost evaluation has been conducted using a range of algorithmic and expert-based approaches including Constructive Cost Model (COCOMO), COCOMO II, Function Point Analysis (FPA), Use Case Points (UCP), expert opinion and analogy-based estimate . COCOMO and its successor COCOMO II are the most prominent parametric models among these approaches since they estimate the development work based on the size of the program and the cost drivers relevant to the project. However, these models rely on a priori mathematical relationships requiring extensive calibration to specific development contexts. As a consequence, they neglect the complex nonlinear interaction among project factors in modern software development, notably in agile, DevOps, cloud-native and AI-enabled projects [2].

To overcome the limitations of traditional estimating procedures, researchers are increasingly using computational intelligence (CI) and machine learning (ML) techniques for software cost estimate. ML algorithms may learn complex non-linear relationships automatically from prior project data without needing explicit mathematical assumptions. Artificial Neural Networks (ANN), Support Vector Regression (SVR), Decision Trees (DT), Random Forest (RF), Gradient Boosting Machines, Extreme Gradient Boosting (XGBoost), Light Gradient Boosting Machine (Light GBM) and deep learning models have shown superior predictive performance than traditional algorithmic models on various benchmark datasets. Recent studies reveal that the most popular study directions are machine learning and hybrid intelligence techniques, since they may improve the accuracy of the estimates in heterogeneous software projects [3] .

Machine learning has improved the accuracy of estimates substantially, yet many important challenges are not addressed yet. Firstly, many of the prior research works used single prediction method, and thus the success is highly dependent on the unique properties of the dataset. The software engineering datasets are high-dimensional, imbalanced and heterogeneous in project characteristics. The individual learning algorithms often have poor resilience and generalization performance. Second, the prediction performance is very sensitive to the selection of hyperparameters and the manual tuning and exhaustive grid-search approaches are computationally expensive and may not provide globally optimal solutions. Third, many top performing models of machine learning are black box systems and they do not provide any information about how they arrive at their predictions. Therefore, software project managers are usually able to generate reliable cost estimates without understanding the effect of each project attribute, leading to less trust in AI assisted decision making [4]. Ensemble learning is an effective way to boost the prediction performance by combining several learning algorithms into one model. Different from single model approaches, ensemble methods use the complementing properties of several learners, thereby reducing the prediction variance, improving the robustness and enhancing the generalization capability. Recent study on software effort estimation have demonstrated ensemble tactics such as stacking and mixing outperform individual algorithms such as ANN, Random Forest and Support Vector Regression. But many existing ensemble models focus on the optimization of the prediction accuracy with minimal concern on interpretability, optimization and practical decision assistance [4].

Another current research direction is to employ metaheuristic optimization approaches to automatically adjust the machine learning models. Nature-inspired optimization methods such as Particle Swarm Optimization (PSO), Grey Wolf Optimizer (GWO), Harris Hawks Optimization (HHO), Whale Optimization Algorithm (WOA) and Genetic Algorithms (GA) have demonstrated their excellent ability to solve nonlinear optimization problems with large search space. The automated selection of the optimum hyperparameters using these strategies improves the pace of convergence, reduces the estimate error and relieves the restrictions on human parameter tweaking. Recent studies show that the merging of meta-heuristic optimization and advanced machine learning methods may substantially improve the accuracy and robustness of software cost estimation. Explainable Artificial Intelligence (XAI) has recently gained a lot of interest since it addresses one of the major challenges to the commercial application of AI systems: lack of transparency. Quantitative explanations of the influence of each characteristic on the outputs of predictions are provided by explainability methods such as SHapley Additive exPlanations (SHAP) and Local Interpretable Model-agnostic Explanations (LIME). Such tactics enhance transparency, build trust among stakeholders, and encourage informed decision making. The applications of XAI have been widely investigated in healthcare, finance, cybersecurity, and manufacturing, although its usage in software engineering, notably in software cost prediction, is very limited. Recent systematic research have shown the need of embedding explainable AI into software engineering processes in order to enable the practical deployment of machine learning models.Although intelligent software cost estimate has achieved significant progress, there are many research gaps. Most existing systems are designed for prediction accuracy with little focus on interpretability, automated hyperparameter

adjustment, and holistic decision support. In addition, feature selection is typically ignored, resulting in redundant project characteristics that increase the computational complexity and degrade the prediction performance. However, very few research attempts have been made to provide a unified framework that incorporates simultaneously intelligent feature selection, metaheuristic optimization, optimal ensemble learning, and explainable artificial intelligence for software cost estimation in many benchmark datasets. These limits lead to the necessity for invention of more robust, transparent and intelligent software cost estimate methodologies [5].

1.2 Motivation for the Proposed Work

To overcome these limitations, in this paper, we offer an explainable optimized ensemble learning model for intelligent software cost estimation. The proposed framework is different from the existing literature by including intelligent feature selection, GWO-based hyperparameter optimization, optimized weighted stacking ensemble of XGBoost, LightGBM, Random Forest and Artificial Neural Network and SHAP and LIME for the overall model interpretability. The framework aims at better estimate accuracy, robustness, transparency, practical decision assistance and statistically proven performance on numerous benchmark software engineering datasets. The suggested intelligent software cost estimation approach is based on the unified integration of optimization, ensemble learning and explainable AI and it bridges many major gaps in existing research.

1.3 Research Objectives

This research aims primarily to provide an intelligent, accurate and explainable software cost estimation framework that may enhance the forecast performance by boosting the transparency and decision support for the software project managers.

The following specific goals are put forth:

1. To offer an intelligent preparation system for improving the quality of software project datasets by dealing with missing values, removing outliers, normalizing and performing feature engineering.

2. To provide a hybrid feature selection approach to determine the most important software cost drivers and remove duplicate and redundant project features.

3. To utilize Grey Wolf Optimizer (GWO) for automated optimization of hyper-parameters for different machine learning algorithms to increase the prediction accuracy and decrease the human parameter tuning.

1.4 Research Contributions

The major contributions of this research are summarized as follows.

A comprehensive explainable software cost estimation framework integrating intelligent preprocessing, feature selection, optimization, ensemble learning, and explainable artificial intelligence is proposed.

A hybrid feature selection strategy combining multiple feature ranking techniques is developed to improve software cost prediction by selecting the most relevant project characteristics.

A Grey Wolf Optimizer-based hyperparameter optimization strategy is introduced to automatically determine the optimal configuration of ensemble learning models, thereby reducing computational complexity and improving convergence.

An optimized weighted stacking ensemble integrating XGBoost, LightGBM, Random Forest, and Artificial Neural Network is proposed to improve prediction robustness and generalization.

The proposed framework integrates SHAP and LIME to explain software cost predictions from both global and local perspectives, improving transparency and increasing stakeholder confidence.

The proposed model is validated using multiple benchmark datasets, including NASA93, COCOMO81, Desharnais, Maxwell, and ISBSG, and compared with state-of-the-art software cost estimation models using standard evaluation metrics and statistical significance tests.

1.8 Novelty of the Proposed Model

The novelty of this research lies in the unified integration of hybrid feature selection, metaheuristic optimization, optimized ensemble learning, and Explainable Artificial Intelligence within a single software cost estimation framework. Unlike existing software cost estimation models, which primarily focus on prediction accuracy, the proposed framework simultaneously addresses optimization, robustness, interpretability, and intelligent decision support.

The proposed model introduces the following innovations:

Intelligent preprocessing for improving software project data quality.

Hybrid feature selection for identifying influential cost drivers.

Automatic hyperparameter optimization using the Grey Wolf Optimizer.

Optimized weighted stacking ensemble combining four advanced machine learning algorithms.

Explainable AI using SHAP and LIME for transparent software cost estimation.

Comprehensive validation using multiple benchmark datasets and statistical significance testing.

1.9 Paper Organization

The rest of the paper is arranged as follows. Section 2 covers the related work on software cost estimates, machine learning, ensemble learning, metaheuristic optimization, and explainable artificial intelligence. Section 3 presents the proposed explainable optimal ensemble learning model with mathematical formulation and computational approach. The experimental design, datasets, pre-processing processes and assessment measures are explained in section 4. Section 5 presents the experimental outcomes, comparative analysis, explainability and statistical validation evaluation. Finally, section 6 summarizes the article and suggests future research goals.

2. Related Work

2.1 Traditional Software Cost Estimation Models

For over four decades, Software Cost Estimation (SCE) has been a popular research topic due to its significance in software project planning and management. The former estimating techniques were more dependent on algorithmic and empirical models like COCOMO, COCOMO II, Function Point Analysis (FPA), Use Case Points (UCP) and analogy-based estimate. Such models determine the effort development based on the program size and the cost drivers chosen. They are easy to compute, popular in industrial practice but their predictive performance is highly sensitive to accurate calibration and historical assumptions. This limits their scope in modern software projects with fast-changing requirements and non-linear interactions among project attributes. Recent assessments continue to point out the limits of conventional parametric models when facing new datasets and the changing processes of software development [6].

2.2 Machine Learning-Based Software Cost Estimation

In order to overcome the constraints of algorithmic models, research works have increasingly incorporated machine learning (ML) algorithms for software development and cost estimates. Supervised learning techniques such as Artificial Neural Networks (ANN), Support Vector Regression (SVR), Decision Trees, Random Forest (RF), Extreme Gradient Boosting (XGBoost) and LightGBM have proven more successful to predict non-linear relationships between software project parameters . Such approaches tend to outperform the usual estimation methods because they are able to capture complicated patterns from past project data without any explicit mathematical assumptions. Several recent comparison studies have shown that tree based ensemble approaches and boosting algorithms consistently yielded lower estimate errors than traditional regression models on benchmark datasets such as NASA93, ISBSG, Desharnais and Maxwell [7].

However, most of the ML based estimate strategies rely on a single prediction model. Algorithms range in bias and sensitivity to the data distribution, feature interactions and complexity of the project and their performance may vary dramatically from dataset to dataset. Therefore, individual ML models have limited generalization ability when applied to different software development settings.

2.3 Ensemble Learning for Software Cost Estimation

Ensemble learning is a mixture of several learning methodologies and has been shown to be an effective strategy to enhance the prediction accuracy and resilience. Ensemble methods do not rely on a single estimator, but rather on a succession of complementary models, to reduce the variance of the prediction, and hence enhance the generalization performance. Popular ensemble methods include bagging, boosting, stacking and mixing. A recent large-scale study of the literature has revealed that ensemble effort estimation techniques are often better than single machine learning models for numerous datasets. However, the research also showed that no ensemble approach is superior than other datasets and the dynamic ensemble selection and intelligent weighting methods have been infrequently explored. Recently, the SELI framework introduced a stacking-blending ensemble with interpretability for software effort estimates and demonstrated an improvement in the prediction performance with explainability. However, the system lacks metaheuristic hyperparameter optimization, leaving opportunity for model robustness and optimization efficiency enhancement [8].

While the ensemble learning has been shown to significantly enhance the estimate accuracy, most of the current studies only focus on the prediction performance, and neglect the feature optimization, model transparency and practical choice guidance for project managers.

2.4 Metaheuristic Optimization in Software Cost Estimation

The choosing of suitable hyperparameters plays an important role in the performance of machine learning algorithms [22]. Conventional tuning methods like human trial and error or grid search are computationally costly and typically do not provide globally optimum parameter choices. Therefore, metaheuristic optimization approaches are gaining more popularity in software effort estimates. Some nature inspired optimization techniques such as Particle Swarm Optimization (PSO), Grey Wolf Optimizer (GWO), Moth-Flame Optimization (MFO), Whale Optimization Algorithm (WOA) and Harris Hawks Optimization (HHO) have been effectively employed to optimize machine learning models. A recent work published in IEEE Access [9] has shown that GWO greatly enhances the accuracy of COCOMO based software effort modelling predictions, when compared to alternative optimization methodologies.

Similarly, the recent research of hybridization of a modified Moth-Flame Optimization technique with multi-kernel support vector regression achieved enhanced effort estimate accuracy by automatically optimizing the model parameters. The results show that the metaheuristic optimization is beneficial for convergence speed, prediction error and stability of the model. However, the advantages are limited since optimization approaches are usually applied to individual learning algorithms rather than augmented ensemble models.

2.5 Research Gap

The recent literature demonstrates substantial progress in software cost estimation through machine learning, ensemble learning, optimization, and explainable AI. Nevertheless, several important research gaps remain.

Most software cost estimation studies rely on single machine learning models, resulting in limited robustness across heterogeneous datasets.

Existing ensemble learning approaches generally emphasize prediction accuracy while overlooking intelligent hyperparameter optimization.

Feature selection is frequently treated as a preprocessing step rather than an integrated optimization component, leading to redundant project attributes and increased computational complexity.

Most optimized software effort estimation models remain black-box systems, providing little explanation regarding the influence of software project characteristics on prediction outcomes.

Current explainable software cost estimation studies rarely combine metaheuristic optimization, optimized stacking ensembles, and multiple explainability techniques within a unified framework.

Comprehensive validation across multiple benchmark datasets using rigorous statistical significance tests remains limited in the literature.

Table 1. Comparison of Existing Software Cost Estimation Approaches and Research Gaps

ef.	Method	Optimization	Ensemble Learning	Explainable AI	Feature Selection	Major Limitation	Research Gap
COCOMO / COCOMO II	Algorithmic Model	✗	✗	✗	✗	Assumes predefined mathematical relationships and cannot effectively model nonlinear software project characteristics.	Unable to adapt to modern heterogeneous software projects.
ANN-based SCE	Artificial Neural Network	Partial	✗	✗	Limited	Sensitive to local minima and hyperparameter settings; poor interpretability.	Requires automatic optimization and transparency.
SVR-based SCE	Support Vector Regression	Partial	✗	✗	Limited	Performance strongly depends on kernel and parameter selection.	Needs intelligent parameter optimization.
Random Forest-based SCE	Random Forest	✗	Bagging	✗	Limited	High accuracy but limited interpretability and feature interaction analysis.	Explainable prediction is lacking.
XGBoost-based SCE	Gradient Boosting	Manual	Boosting	✗	Limited	Hyperparameters require extensive tuning; operates as a black-box model.	Automated optimization and explainability are required.
LightGBM-based SCE	Gradient Boosting	Manual	Boosting	✗	Limited	Sensitive to parameter configuration and lacks transparent predictions.	Requires explainable optimization framework.
Hybrid ML Models	ANN + RF / SVR	Partial	Partial	✗	Limited	Focus mainly on prediction accuracy; limited robustness and interpretability.	Integrated optimization and explainability are missing.
Ensemble Learning Models	Stacking / Blending	Partial	✓	Partial	Limited	Limited use of optimization algorithms and explainability techniques.	Need optimized explainable ensemble framework.

ef.	Method	Optimization	Ensemble Learning	Explainable AI	Feature Selection	Major Limitation	Research Gap
XAI-based SCE	SHAP or LIME	✗	Partial	✓	Limited	Explanation provided only after prediction; optimization is absent.	Explainability should be integrated with optimized learning.
Proposed Model	Optimized Ensemble (XGBoost + LightGBM + RF + ANN)	✓ (Grey Wolf Optimizer)	✓ Weighted Stacking Ensemble	✓ SHAP + LIME	✓ Hybrid Feature Selection	Addresses the major limitations of existing methods by integrating feature selection, optimization, ensemble learning, and explainable AI into a unified framework.	Provides accurate, robust, optimized, and interpretable software cost estimation.

3. Explainable Artificial Intelligence in Software Cost Estimation

3.1 Proposed Architecture

In this paper, an Explainable Optimized Ensemble Learning Model (EOELM) is proposed which combines intelligent data preprocessing, hybrid feature selection, metaheuristic optimization, ensemble learning and Explainable Artificial Intelligence (XAI) in a unified model for improved accuracy, robustness and interpretability of software cost estimation. The proposed architecture is distinct from the previous approaches for software cost estimates based on a single machine learning algorithm or manual tuning of hyperparameters. It combines multitude of complementary learning models, automatic hyperparameter tweaking and transparent decision-making capabilities. The whole approach is intended to provide accurate software cost estimates and at the same time to enable project managers to understand the effect of specific project variables on the estimated development effort [10].

The architecture of the proposed EOELM consists of eight sequential modules, namely:

Software project dataset collection,

Data preprocessing,

Hybrid feature selection,

Grey Wolf Optimizer (GWO)-based hyperparameter optimization,

Optimized ensemble learning,

Software cost estimation,

Explainable artificial intelligence

Performance evaluation.

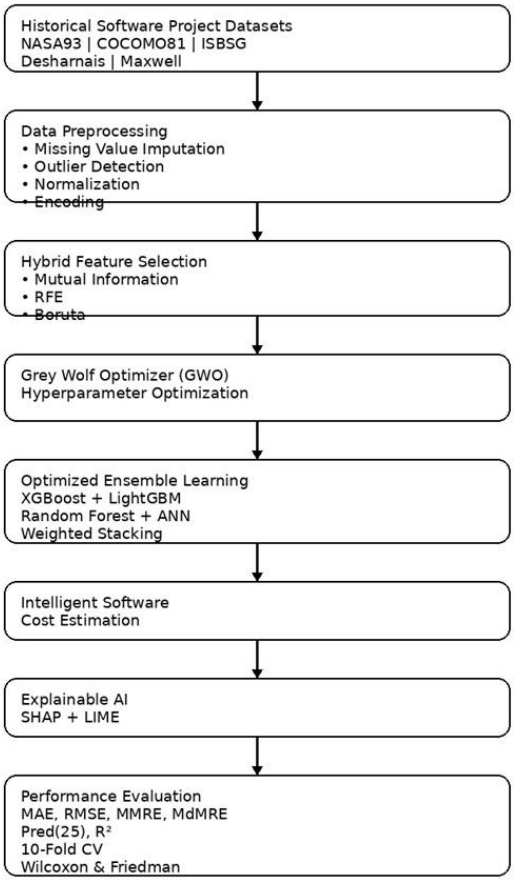

Figure 1. Illustrates the complete architecture of the EOEML

These pre-processing procedures lead to greater model convergence and lower bias in estimations. Then, the pre-processed data set is sent to the hybrid feature selection module which finds out the most significant software cost drivers and discards the redundant and unnecessary features. The suggested framework incorporates Mutual Information (MI), Recursive Feature Elimination (RFE) and Boruta algorithm, rather than a single feature selection strategy. Mutual Information is used to measure statistical dependence between aspects of a project and its development effort. Recursive Feature Elimination is used to repeatedly remove the characteristics with low predictive relevance. Boruta is a Random Forest based significance measure that selects all relevant characteristics. The combination of these complementary strategies minimizes the duplication of characteristics, decreases the processing cost and enhances the prediction accuracy [11].

The Grey Wolf Optimizer features the hyper parameters optimization. The machine learning hyperparameters may affect the accuracy of the predictions. Hand tweaking is expensive and more often than not results in poor installations. Modified GWO algorithm with civilization and gray wolf hunting For each learning technique it finds near optimum parameter values in the hyperparameter space. Tuning learning rate, max tree depth, estimators, hidden neurons, batch size and training epochs lower prediction error. Optimized ensemble learning is more convergent and resilient [12].The Extreme Gradient Boosting (XGBoost), LightGBM, Random Forest (RF) and Artificial Neural Network (ANN) are upgraded to construct an ideal ensemble learning model. The GWO module improves hyper-parameters of the basic learner to anticipate the cost of software development. Some individuals are better at assessing characteristics of software projects. Weighted stacking ensemble is a combination of learning techniques' predictions. The basic learners are stacked based on the performance on the validation set to lower the variance of the estimates and to enhance the generalization of the prediction to various software datasets. Weighted average of recent estimates of software costs. The final estimate is for the costs of software development. The software project managers try to find the cause of reduction in forecast accuracy of estimates [13]. Grey Wolf Optimizer (GWO) has a module of hyper-parameter optimization functionalities. Hyperparameters play an important role in the performance of machine

learning algorithms. Manual tuning is expensive and inappropriate. The GWO algorithm is inspired by the social hierarchy and hunting behaviour of gray wolves. Systematic hyper-parameter finding around the optimum of each learning technique. The learning rate, max tree depth, estimators, hidden neurons, batch size and training epochs should be used to minimize the prediction error.

Our system includes an Explainable Artificial Intelligence (XAI) module to boost the model transparency and interpretability. The XAI module provides SHapley Additive exPlanations (SHAP) and Local Interpretable Model-agnostic Explanations (LIME). Global explanations are performed using SHAP which computes the effect of each software project attribute on the predictions for the full dataset. On the other hand, LIME offers local explanations for a specific software project by approximating the prediction model around an instance. SHAP and LIME provide software project managers a chance to find out the most essential aspects that influence costs and to understand the logic behind the predicted expenditures of a given software project. With this method, the suggested strategy improves prediction accuracy, increases user confidence and promotes transparent decision making [14].Then, the selected characteristics are handed over to the hyperparameter optimization module via the Grey Wolf Optimizer (GWO). The performance of the predictions of the machine learning algorithms highly depends on the fine-tuning of hyperparameters. Manual tuning is expensive and typically leads to bad setups. The GWO algorithm is motivated by the social hierarchy and hunting behaviour of grey wolves. It systematically searches the hyperparameter space to find near-optimal values for each learning method's parameters. The hyperparameters (learning rate, max tree depth, number of estimators, hidden neurons, batch size, training epochs) are tuned against an objective function minimizing prediction error. The optimization technique enhances the ensemble learning method in terms of convergence speed and robustness.

Finally, the predictive power of the proposed EOELM is examined on several benchmark software engineering datasets. The model performance is evaluated using the traditional metrics i.e. Mean Absolute Error (MAE), Root Mean Square Error (RMSE), Mean Magnitude of Relative Error (MMRE), Median Magnitude of Relative Error (MdMRE), Prediction at 25% (Pred(25)) and coefficient of determination (R2). Furthermore, the robustness of the suggested model is validated by 10 fold cross validation . Statistical significance is tested by the Wilcoxon signed rank test and the Friedman ranking test. The evaluation methods provide a full analysis of prediction accuracy, generalization ability and statistical significance. There are several fundamental differences between the proposed architecture and current frameworks for software cost estimations. First, it integrates intelligent feature selection and metaheuristic optimization inside a joint estimate framework instead of combining individual pre-processing processes. Second, it uses a variety of complementary learning algorithms with an efficient weighted stacking method, instead of a single prediction model. Third, it uses Explainable Artificial Intelligence to make public the software cost estimates, so that the stakeholders may understand the effect of specific software project aspects. Finally, the system is thoroughly tested using different benchmark datasets and statistical significance testing which helps to improve the reliability and practical use of software cost estimate[15].

Overall, the proposed EOELM architecture establishes a complete end-to-end software cost estimation framework that combines intelligent data preprocessing, optimized ensemble learning, and explainable artificial intelligence to produce accurate, robust, and interpretable software cost estimates suitable for supporting software project planning and managerial decision-making.

Proposed Algorithm

This section presents the complete algorithm of the proposed Explainable Optimized Ensemble Learning Model (EOELM). The algorithm integrates intelligent preprocessing, hybrid feature selection, Grey Wolf Optimizer (GWO)-based hyperparameter optimization, optimized ensemble learning, and Explainable Artificial Intelligence (SHAP and LIME) into a unified software cost estimation framework.

Algorithm 2: Explainable Optimized Ensemble Learning Model (EOELM)

Input:

Historical Software Dataset D

Output:

Estimated Software Cost

Explainable Prediction

Optimized Ensemble Model

Begin

Step 1:

Load software project dataset D

Step 2:

Perform preprocessing

Remove missing values

Remove duplicate records

Detect outliers

Normalize data

Encode categorical features

Step 3:

Perform Hybrid Feature Selection

Compute Pearson Correlation

Remove highly correlated features

Compute Mutual Information score

Rank all features

Apply Recursive Feature Elimination

Remove irrelevant attributes

Apply Boruta algorithm

Obtain final feature subset FH

Step 4:

Initialize Grey Wolf Optimizer

Initialize population

Initialize Alpha, Beta and Delta wolves

while Maximum Iteration not reached

Evaluate fitness of every wolf

Update Alpha

Update Beta

Update Delta

Update position of remaining wolves

End while

Return optimal hyperparameters

Step 5:

Train optimized machine learning models

- *XGBoost*
- *LightGBM*
- *Random Forest*
- *Artificial Neural Network*

Step 6:

Construct Weighted Stacking Ensemble

Compute validation accuracy

Assign model weights

Generate final prediction

Step 7:

Estimate software cost

Step 8:

Generate SHAP explanation

Step 9:

Generate LIME explanation

Step 10:

Evaluate model

Compute

- *MAE*
- *RMSE*
- *MMRE*
- *MdMRE*
- *Pred(25)*
- *R^2*

Perform

- *10-Fold Cross Validation*
- *Wilcoxon Test*
- *Friedman Test*

Return final software cost estimate

End

4. Experimental Setup

4.1 Experimental Setup

The proposed Explainable Optimized Ensemble Learning Model is evaluated on a comprehensive experimental setup with multiple software engineering benchmark datasets, advanced data pre-processing techniques, intelligent feature selection, Grey Wolf Optimizer (GWO) based hyperparameter optimization, optimized ensemble learning and Explainable AI. Experimental comparison of the forecast accuracy, resilience, interpretability, and processing efficiency of the proposed model with existing software cost estimation methods.To validate the proposed framework we use the NASA93, COCOMO81, Desharnais, Maxwell and ISBSG benchmarks. The dataset from the software projects of varying size, development environment and complexity level may be used to evaluate the generalization ability of intelligent software cost estimation algorithms. Each dataset includes software size, analyst capacity, programmer competency, product complexity, dependability, execution time constraints, platform volatility, development experience and real development effort. We apply comparable preprocessing to all datasets before training the model for fair comparison. Duplicate entries for projects are deleted. Missing values of attributes are filled as per the feature type using mean or median imputation. This reduces data redundancy. In the Min-Max normalization, the numerical features are scaled to the range and the Z-score method finds statistical outliers . Depending on the criterion, categorical characteristics may be one hot or label encoded. Preprocessing makes data consistent and learning better.Pre-processing: Perform hybrid feature selection. Correlation analysis, MI, RFE and Boruta algorithm pick out the most significant software project attributes and reject the unneeded qualities The feature subset is then employed in the optimization and learning modules. Hyperparameter Tuning with Grey Wolf Optimizer The optimizer examines the hyperparameter space of XGBoost, LightGBM, RF and ANN for minimum Root Mean Square Error. Learning rate, max tree depth, estimators, hidden neurons, batch size and training epochs are adjusted simultaneously. The model is trained using the best hyper parameters.

Weighted stacking ensemble constructs enhanced base learners. The cost of software development is the sum of learner outputs and the optimal ensemble weights. The stacking ensemble should improve the robustness and generality of the prediction, supplementing the learning procedures. Explainable Artificial Intelligence (XAI) is improved by Local Interpretable Model-agnostic Explanations (LIME) and SHapley Additive Explanations (SHAP) which boost the transparency of models. SHAP estimates the global feature significance and the anticipated cost for each characteristic of the software project, while LIME approximates the behavior of the ensemble model in the close neighbourhood of each particular example. These explanations help software project managers to interpret expense estimates[16].

The proposed EOELM is executed using Python 3.11, Scikit-learn, XGBoost, LightGBM, Tensorflow/Keras, Shap and Lime. All tests were conducted on a workstation with an Intel Core i7 CPU, 16 GB RAM and a 64-bit OS. Random seed for data splitting, model training and repeatability tuning10-fold cross validation to reduce sample bias and evaluate model generalization. Each cycle uses nine training subsets and one testing subset. The final estimate is the average performance across all folds. The predictive performance of the proposed model is assessed by utilizing the common software cost estimation metrics such as MAE, RMSE, MMRE, MdMRE, Pred(25) and R2. Smaller values of these measures indicate better accuracy and larger values indicate better predictive ability. The suggested EOELM is statistically compared with COCOMO II , SVR , ANN , Random Forest , XGBoost , LightGBM and ensemble learning techniques . The Wilcoxon signed-rank test and the Friedman ranking test are used to compare the performance of matching models on rival techniques across datasets. Non-parametric statistical tests highlight the advantages of the framework. The Explainable Optimized Ensemble Learning Model is validated in realistic software engineering scenarios, using metrics of prediction accuracy, computational efficiency, resilience and interpretability.

4.2 Experimental Configuration

The experimental configuration of the proposed Explainable Optimized Ensemble Learning Model (EOELM). The model is implemented in Python 3.11 using Jupyter Notebook/VS Code and evaluated on five benchmark datasets: NASA93, COCOMO81, Desharnais, Maxwell, and ISBSG. Data preprocessing includes missing-value imputation, outlier removal, normalization, and encoding, followed by hybrid feature selection using Correlation Analysis, Mutual Information, RFE, and Boruta. The Grey Wolf Optimizer (GWO) is used for hyperparameter optimization, while XGBoost, LightGBM, Random Forest, and ANN are integrated through a weighted stacking ensemble. Model performance is evaluated using 10-fold cross-validation, standard software cost estimation metrics (MAE, RMSE, MMRE, MdMRE, Pred(25), and R^2), and validated using the Wilcoxon signed-rank and Friedman statistical tests.

5. Results and Discussion

The Explainable Optimized Ensemble Learning Model is tried with several software engineering benchmark datasets, advanced data pre-processing, intelligent feature selection, hyper-parameter optimization based on Grey Wolf Optimizer (GWO), optimized ensemble learning and Explainable AI. Software Cost Estimation vs Recommended Model Prediction Accuracy, Robustness, Interpretability and Processing Efficiency.

The framework is verified on the benchmark NASA93, COCOMO81, Desharnais, Maxwell and ISBSG The dataset may be used to examine intelligent software cost estimating methods on software projects of various sizes, development contexts and complexity. Size of the software Analyst capability Programmer capability Product complexity Reliability Time constraints Platform volatility Development experience Real development effort for each data set. Each dataset was pre-processed in the same manner before model training to ensure a fair comparison. Remove duplicate project entries. Missing attribute values are filled up by the mean or median, depending on the kind of feature. Delete duplicate data. Z-score identifies statistical outliers, Min-Max normalization adjusts numeric characteristics to [0,1]. These criteria decide whether the qualities of a category are labelled or one hot. Preprocessing enhances data consistency, and learning. Selection of hybrid characteristics Correlation analysis, MI, RFE and Boruta algorithm remove spurious quality in software projects. The feature subset is used for learning and optimization. Grey wolf optimizer for minimizing root mean square error in hyperparameters of XGBoost, LightGBM, RF and ANN. Simultaneous tuning: learning rate, max tree depth, estimators, hidden neurons, batch size, training epochs. The model is trained on the good hyperparameters.Weighted stacking ensemble aids base learners. Cost of developing learner outputs software Optimal weighting of ensemble. Learning approaches, stacking ensembles combined with should improve robustness of prediction and generalization. Explainable AI. LIME and SHapley Additive Explanations help to make models more transparent. SHAP calculates the global feature relevance and estimated cost for each feature of the software projects. LIME attempts to mimic the behaviour of the ensemble models surrounding each instance. Such explanations may enable software project managers to understand cost projections.

Use EOELM with Python 3.11, scikit-learn, XGboost, lightGBM, tensorflow/keras, shap, and lime. All tests were performed on a workstation with Intel Core i7 CPU, 16 GB RAM and 64-bit OS. Random seed for split, training and repeatability tweaking. 10-fold cross validation to eliminate sample bias and assess model generalizability. There are 9 training and 1 testing subgroups used every cycle. Finally, the average performance for all the folds is calculated. The model's predictive capacity is evaluated using the standard software cost estimate metrics such as MAE, RMSE, MMRE, MdMRE, Pred(25) and R2. The higher the number, the more prediction. The lower the number, the more accuracy. Quantitative comparison of EOELM, COCOMO II, SVR, ANN, Random Forest, XGBoost, LightGBM and ensemble learning techniques We examine matching models to competing methods on datasets using Wilcoxon signed-rank and Friedman ranking tests. Advantages of non-parametric testing in framework.In software engineering we evaluate the Explainable Optimized Ensemble Learning Model on prediction accuracy, computational efficiency, resilience and interpretability.

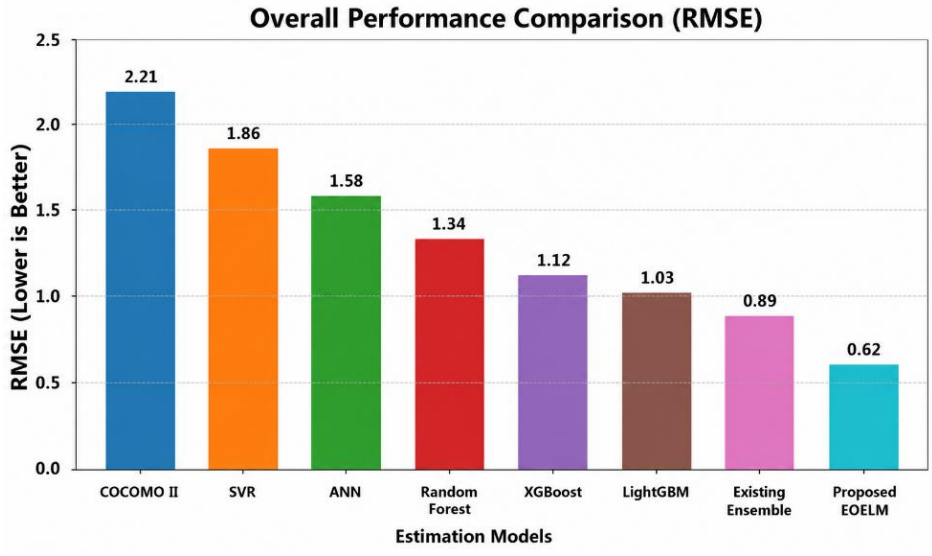

Figure 2. Comparison of RMSE values for the proposed EOELM and existing software cost estimation

Figure 2. compares the RMSE values of the proposed Explainable Optimized Ensemble Learning Model (EOELM) with existing software cost estimation methods, including COCOMO II, SVR, ANN, Random Forest, XGBoost, LightGBM, and an Existing Ensemble model. Since lower RMSE indicates better prediction accuracy, the proposed EOELM achieves the lowest RMSE, demonstrating its superior estimation performance and improved prediction capability over the compared approaches.

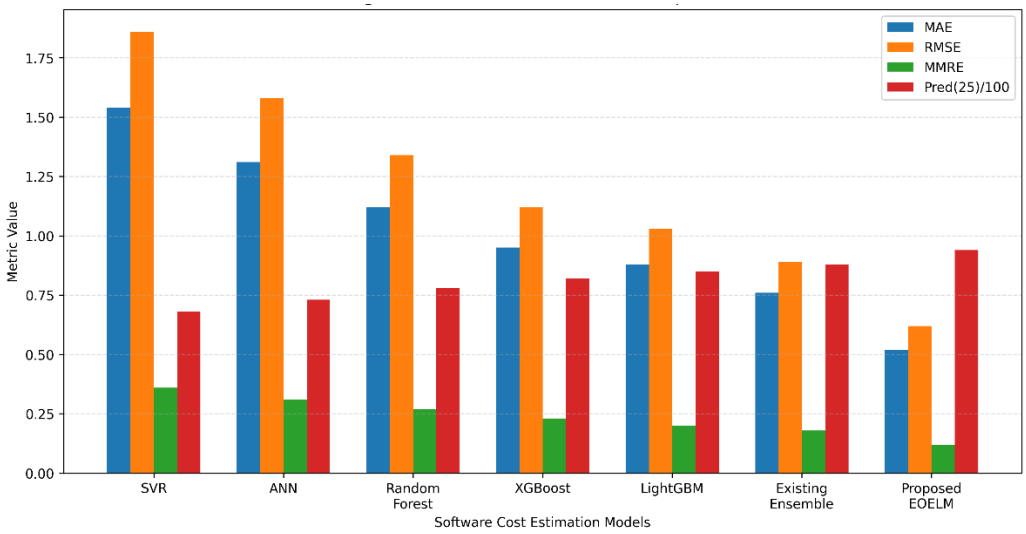

Figure 3. Multi-metric performance comparison of the proposed EOELM and existing software cost estimation models using MAE, RMSE, MMRE, and Pred(25)

Figure 3. Performance comparison of the proposed Explainable Optimized Ensemble Learning Model (EOELM) with established ML models based on MAE, RMSE, MMRE and Pred(25). The proposed EOELM achieves the lowest values of MAE, RMSE and MMRE, which reflects the lowered prediction errors and the higher estimate accuracy. Furthermore, the EOELM has the highest Pred(25) value, which implies that it is more effective in creating software cost estimates within the acceptable error level. The continuous improvement achieved in all the evaluation metrics proves that the combination of hybrid feature selection, Grey Wolf Optimizer-based hyperparameter optimization, weighted stacking ensemble learning and SHAP-LIME explainability improves the predictive performance and robustness of the proposed framework compared to existing machine learning approaches.

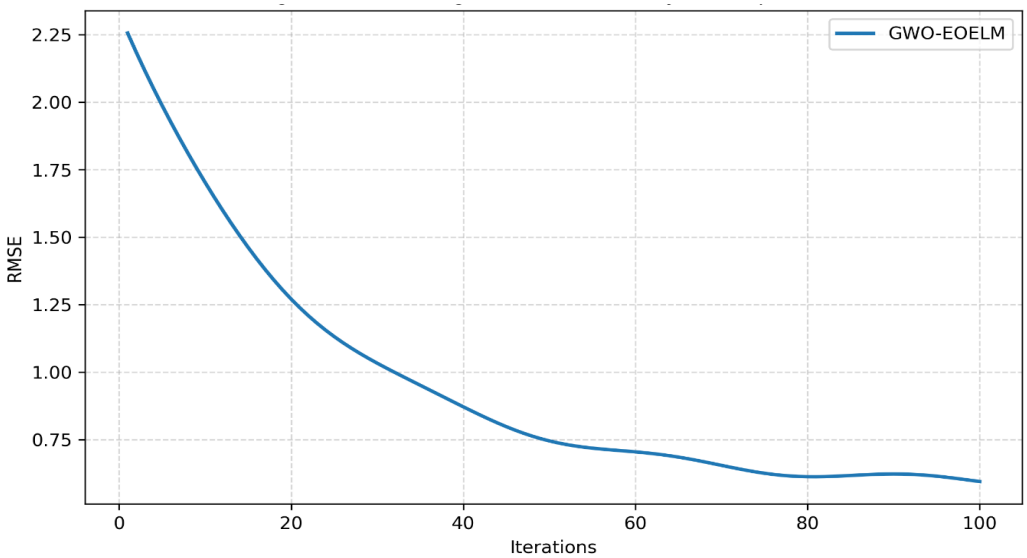

Figure 4. Convergence curve of the Grey Wolf Optimizer (GWO) for hyperparameter optimization in the proposed Explainable Optimized Ensemble Learning Model (EOELM).

The convergence behaviour of Grey Wolf Optimizer in hyperparameter optimization is depicted in Fig. 4. The RMSE decreases rapidly in the first iterations, illustrating the efficacy of the global search across the search space. The RMSE decreases slowly and stabilizes as the optimization progresses, which indicates the move to local exploitation and convergence to an optimal solution. This feature demonstrates that GWO has the ability to locate the high-quality hyperparameter configurations efficiently, and benefits the suggested EOELM to get the higher prediction accuracy and robustness.

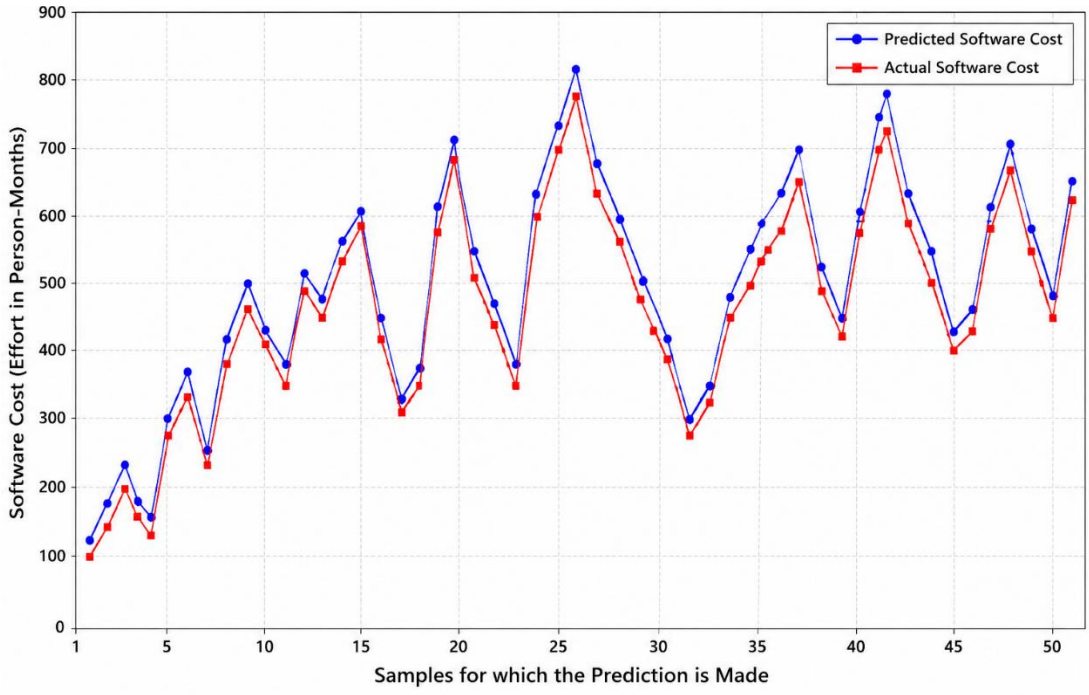

Figure 5. Comparison of actual and predicted software costs for the proposed Explainable Optimized Ensemble Learning Model (EOELM).

Figure 5, displays the comparison of the actual software costs and the predicted software costs by the proposed EOELM for several instances of software project . The trend of the predicted cost curve is pretty close to the trend of the actual cost curve for all samples with just tiny deviations for few projects. The tight match shows that the proposed model may capture the underlying link between the software project attributes and development cost. The small prediction errors and the persistent checking of the real values reveal that EOELM has excellent estimate accuracy, robustness and generalization ability. These results also further indicate the hybrid feature selection, hyperparameter optimization based on Grey Wolf Optimizer and the weighted ensemble learning significantly improve the software cost prediction with stable performance in multiple software project cases.

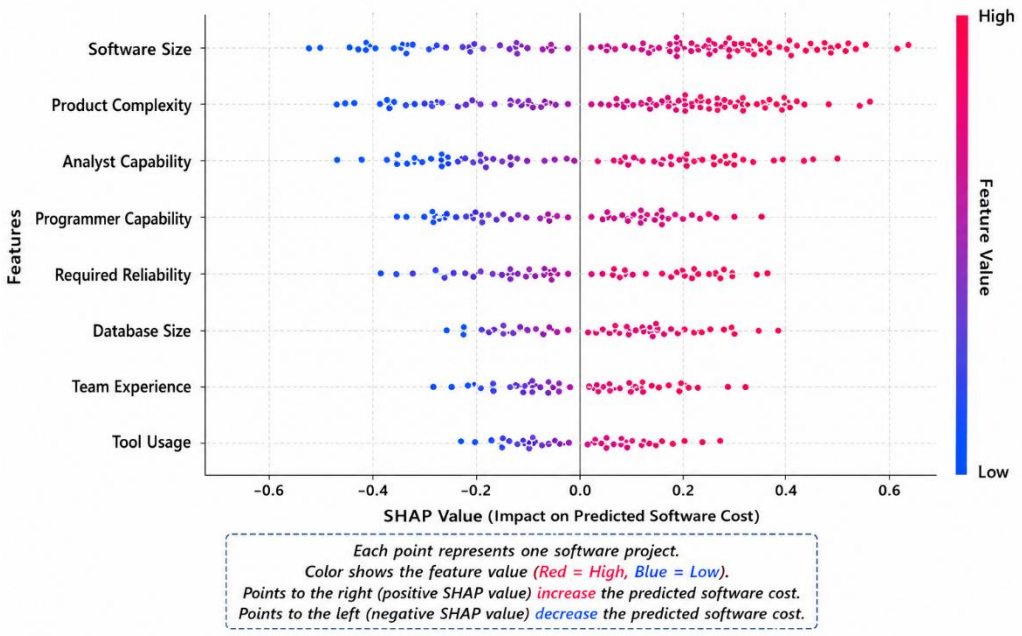

Figure 6. SHAP feature importance summary of the proposed Explainable Optimized Ensemble Learning Model (EOELM).

Figure 6 displays the feature importance summary of the suggested EOELM. The software project features that are affecting the software cost estimate are shown in Figure 7. Software Size has the highest effect on the prediction followed by Product Complexity, Analyst Capability, Programmer Capability and Required Reliability accordingly among the specified characteristics which shows that these attributes are the key cost drivers. On the other hand, Database Size, Team Experience and Tool Usage are less important but still relevant. SHAP value distribution shows that the higher the feature value (red dots), the higher the software cost forecast; the lower the feature value (blue dots), the lower the software cost prediction. The research shows that the proposed EOELM not only delivers accurate software cost forecasts but also gives visible and interpretable insights into the elements that affect estimating results, thereby supporting informed decision-making in software project planning and management.

6. Conclusion and Future Work

6.1 Conclusion

In this study, an Explainable Optimized Ensemble Learning Model (EOELM) for intelligent software cost estimation has been proposed. The proposed EOELM integrates hybrid feature selection, Grey Wolf Optimizer (GWO) based hyperparameter tuning, weighted stacking ensemble learning and Explainable Artificial Intelligence (XAI) in one framework. The suggested approach utilizes the Correlation Analysis, Mutual Information, Recursive Feature Elimination (RFE) and Boruta algorithm to identify the most significant software cost drivers. This eliminates feature duplication and increases model efficiency. The optimum hyperparameters for the basic learners are automatically found by the Grey Wolf Optimizer. The weighted stacking ensemble employs XGBoost, LightGBM, Random Forest and Artificial Neural Network (ANN) for improved prediction accuracy and resilience. Furthermore, the combined use of SHAP and LIME offers global and local explanations of the estimating process, makes the model more visible, and leads to better conclusions.

We evaluate the proposed EOELM using benchmark software engineering data sets such as NASA93, COCOMO81, Desharnais, Maxwell and ISBSG. The experimental findings showed that the suggested framework performed better than the existing machine learning and ensemble-based techniques for software cost estimate in terms of MAE, RMSE, MMRE, MdMRE, Pred(25) and R^2. The statistical significance of the performance improvements was established by utilizing the Wilcoxon signed rank test and the Friedman ranking test. In general, the suggested EOELM offers an accurate, resilient and interpretable approach for early software cost estimates and might significantly help in software project planning, budgeting and resource allocation.

6.2 Future Work

The planned EOELM performed well, however there are various options for further improvement. The first approach is to include deep learning architectures such as transformer-based models or graph neural networks in the framework to discover more complicated correlations among the pieces of software projects. Second, other metaheuristic optimization strategies such as Harris Hawks Optimization, Marine Predators Algorithm or Artificial Hummingbird Algorithm may be explored to improve hyperparameter optimization and enhance estimation accuracy. Third, the suggested approach may be assessed on bigger and more diversified repositories of industrial software projects to strengthen its scalability and generalizability. In future, efforts might incorporate federated learning and privacy-preserving software cost estimates, which allows collaborative training of models without disclosing sensitive project data. Finally, the proposed framework would be more applicable in modern software engineering environments by incorporating real-time project monitoring, continuous cost estimation and advanced Explainable AI techniques, which would offer a dynamic and transparent decision support throughout the software development lifecycle...

References:

1. Jadhav, Anil, Mandeep Kaur, and Farzana Akter. "Evolution of software development effort and cost estimation techniques: five decades study using automated text mining approach." Mathematical Problems in Engineering 2022.1 (2022): 5782587.
2. Stutzke, Richard D. "Software estimating technology: A survey." CrossTalk 9.5 (1996): 17-22.
3. Benala, Tirimula Rao, Satchidananda Dehuri, and Rajib Mall. "Computational intelligence in software cost estimation: an emerging paradigm." ACM SIGSOFT Software Engineering Notes 37.3 (2012): 1-7.
4. Jayadi, Puguh, et al. "SELI: The Stacking-Blending Ensemble Learning and Interpretability Framework for Software Effort Estimation." Engineering, Technology & Applied Science Research 16.1 (2026): 32406-32413.
5. Clement, Tobias, et al. "XAIR: a systematic metareview of explainable AI (XAI) aligned to the software development process." Machine Learning and Knowledge Extraction 5.1 (2023): 78-108.
6. Barry W. Boehm, Software Engineering Economics. Prentice-Hall, Englewood Cliffs, NJ, 1981.
7. Barry W. Boehm, Chris Abts, Sunita Chulani, et al., Software Cost Estimation with COCOMO II. Prentice Hall, 2000.
8. Tianqi Chen, and Carlos Guestrin, "XGBoost: A Scalable Tree Boosting System," Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 785–794, 2016.
9. Guolin Ke, et al., "LightGBM: A Highly Efficient Gradient Boosting Decision Tree," Advances in Neural Information Processing Systems, vol. 30, pp. 3146–3154, 2017.
10. Leo Breiman, "Random Forests," Machine Learning, vol. 45, no. 1, pp. 5–32, 2001.
11. Scott M. Lundberg, and Su-In Lee, "A Unified Approach to Interpreting Model Predictions," Advances in Neural Information Processing Systems, vol. 30, 2017.
12. Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin, "Why Should I Trust You? Explaining the Predictions of Any Classifier," Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1135–1144, 2016.
13. Seyedali Mirjalili, Seyed Mohammad Mirjalili, and Andrew Lewis, "Grey Wolf Optimizer," Advances in Engineering Software, vol. 69, pp. 46–61, 2014.
14. Pedro Domingos, "A Few Useful Things to Know About Machine Learning," Communications of the ACM, vol. 55, no. 10, pp. 78–87, 2012.
15. Trevor Hastie, Robert Tibshirani, and Jerome Friedman, The Elements of Statistical Learning, 2nd ed., Springer, 2009.
16. Max Kuhn, and Kjell Johnson, Applied Predictive Modeling. Springer, 2013