International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 16 (2024) pp.664–676

Article

A Stacking Ensemble Learning Model for Software Development Cost Estimation

Zaineb Sakhrawi [1],* Taher Labidi [2], Asma Sellami [1] and Nadia Bouassida [1]

[1] MIRACL Laboratory, Higher Institute of Computer Science and Multimedia, University of Sfax, Tunis Road Km 10, B.P. 242, Sfax 3021, Tunisia; asma.sellami@isims.usf.tn (A.S.); nadia.bouassida@isims.usf.tn (N.B.)

[2] MIRACL Laboratory, Higher Institute of Computer Science of Medenine, University of Gabes, Djerba Road Km 3, B.P. 283, Medenine 4100, Tunisia; taherlabidi@gmail.com

* Correspondence author:zaineb.sakhraoui@isims.usf.tn

Received date: 5 March 2024; Accepted date: 21 March 2024; Published online: 10 July 2024

Abstract: Estimating software costs is a vital step in guaranteeing the successful completion of a software project. Given the significant impact of Functional Size (FS) measurement on obtaining accurate estimates for enhancement and development projects efforts, this study aims to investigate the use of FS as the key independent variable for predicting software project development costs. This is accomplished by utilizing ensemble models. The dataset used in this study came from the International Software Benchmarking Standards Group (ISBSG) repository. This research compares various single Machine Learning (ML) models and ensemble models learning using Grid Search (GS) tuning techniques to demonstrate the efficacy of our approach. Using the FS of a new development request, the following observations were made: (i) The suggested Stacking Software Development Cost Estimation (StackSDCE) model outperformed the three distinct ML algorithms applied independently. (ii) The application of GS for fine-tuning and configuring the individual ML methods led to improved precision of the StackSDCE outcomes. (iii) StackSDCE-based GS tuning resulted in more precise estimations.

Keywords: software development cost estimation; functional size; machine learning techniques; grid search tuning technique; stacking ensemble model

1. Introduction

This paper expands upon the conclusions derived from a previous research study [1], which focused on enhancing data quality to achieve heightened accuracy in regression test effort estimation. The previous work gave fundamental insights into the significance of numerous parameters influencing test effort estimation, with a particular emphasis on the crucial function of Functional Size (FS). The findings highlighted FS as a critical driver for improving the accuracy of test effort estimating models. Building on this foundation, the current study aims to expand the use of FS as a key factor, moving from its effect on accuracy test effort estimation to its use as an independent variable for development cost estimation. Using the FS, our objective is to elevate the precision of Software Development Cost Estimation (SDCE) models, thereby facilitating more effective project planning and resource allocation in software development endeavors.

Software cost estimation is defined as a critical step of project management, having profound implications for project success [2]. The accuracy of these estimations has a significant impact on project execution, allowing project managers to predict the resources needed to complete activities within certain schedules and specifications [2]. However, the field of software project estimation is challenging, with overestimation and underestimation being primary concerns. Overestimation poses the risk of financial loss and resource underutilization, while underestimation may precipitate project delays, non-compliance with specifications, or even project failure [2]. According to the CHAOS Report 2015, a statistics 56% of projects exceeded their allocated budgets, 60% missed their specified deadlines, and 44% failed to meet their predefined objectives [2]. To address these issues, improving data quality as the initial step in the estimation process is beneficial.

Cost prediction holds paramount importance for every business as it serves as a precursor for establishing budgetary figures and allocating resources throughout the project life cycle [3, 4]. Acquiring input data for the cost

estimation process can be challenging, particularly when the scope of work is uncertain, potentially resulting in imprecise and rough estimates. The greater the clarity regarding the project scope, the higher the likelihood of producing more accurate estimates, as it allows for a more detailed definition of project specifications.

In recent years, the ensemble learning method has been one of the widely adopted methods in improving the accuracy of software cost/effort estimation [5]. The ensemble learning method combines multiple diverse base models to create the most suitable prediction model [5].

This study investigates the process of estimating the costs involved in software development and evaluates the effectiveness of different models in producing accurate estimations. Specifically, three diverse machine learning (ML) techniques - Neural Networks (NN), Support Vector Regression (SVR), and Decision Tree (DT)—were employed to build StackSDCE models. These methods were chosen for their proven accuracy in predicting the cost/effort required in the Software Development Life Cycle (SDLC) [6, 7, 8, 9, 10, 11].

The models were trained and tested using a dataset from the International Software Benchmarking Standards Group (ISBSG) repository, which encompassed a wide range of functional sizes measured in function points. Functional size measurement is crucial as it significantly influences cost/effort prediction in software projects [9]. Additionally, existing research suggests that the size of functional change requests, particularly enhancements, should serve as the primary independent variable in StackSDCE models [9].

The objectives of this study are as follows:

- Examine the impact of functional size on SDCE.
- Apply the Grid Search (GS) Tuning techniques for tuning the hyperparameters of the three selected ML techniques.
- Implement an ensemble model for constructing a cost development estimation model.

These objectives underscore the comprehensive approach adopted in this study, which aims to explore the relationship between functional size and SDCE while employing advanced tuning techniques and ensemble modeling to enhance the accuracy of cost estimation.

The remainder of the paper is organized as follows: Section 2 discusses the context and related work. Section 3 describes our proposed research process methodology. Section 4 presents and examines the findings of the experiments. Section 5 discusses the concerns about the validity of this study. Section 6 discusses the conclusion and next work.

2. Background and Literature Review

This section gives an overview of the Stacking Ensemble Model (SEM) and three chosen machine learning techniques: NN, SVR, and DT, focusing on their application in software development cost/effort estimation.

2.1 Ensemble Learning

In ensemble learning, multiple base models are combined to create a single best-fit prediction model [12]. This approach has been employed successfully in various supervised and unsupervised learning tasks, including classification, distance learning, and regression. The three common types of ensemble learning methods are stacking, boosting, and bagging. For the present study, a stacking-based ensemble learning model was selected for the following reasons [2, 13, 14]:

- A stacking-based ensemble learning model often involves the consideration of various weak learners and combines at least two different individual methods [12]. These methods are then learned in parallel and combined by training a meta-learner to deliver predictions based on the multiple predictions of the weak learners. In contrast, bagging and boosting utilize similar weak learners and combine similar individual methods [12].
- The stacking model is also referred to a meta-ensemble model [12]. In a meta-ensemble model, a meta-model, such as an ending estimator, is employed to learn how to combine the base models. In contrast, boosting and bagging utilize deterministic algorithms to combine weak learners. In a meta-learner, predictions serve as features, and targets serve as ground truths in the data, enabling the model to learn the optimal way to combine input predictions for more accurate output predictions [12].
- In a stacking-based ensemble learning model, unlike boosting, each lower-level model is trained in parallel. The dataset used to train the subsequent model consists of the predictions made by the lower-level models. This creates a stack where the top layer of the model is more finely tuned than its lower layers. Consequently, the top-layer model exhibits high predictive precision and is constructed based on the predictions of the

lower-level models. The stack continues to grow until the best prediction with the least amount of errors is obtained. Therefore, meta-prediction models rely on the predictions of multiple weak or lower model layers to construct a model with fewer biases [15].

Ensemble learning models have recently attracted attention in the software engineering industry [15, 16]. According to [12], ensemble learning models outperform single-model counterparts. Sakhrawi et al. [16, 17] concluded that the SEM shows promise in improving the accuracy of single models used for estimating effort in both conventional and scrum software development. The use of ensemble learning models has increased in both conventional and agile contexts [18]. Several studies have employed the SEM to estimate effort during the development and maintenance phases exclusively [16, 17]. This study is unique in its use to enhance the accuracy of cost prediction during the software development phase while using FS as the primary independent variable.

2.2 Neural Network

NN stand out as a prevalent choice in software cost/effort estimation (SEE) due to their widespread adoption and effectiveness [19]. They offer a versatile approach that has shown considerable enhancements in accuracy compared to traditional methods [19].

In this investigation, we employed a fully connected multi-layer neural network, commonly referred to a Multilayer Perceptron (MLP). The architecture of the MLP comprises several layers of interconnected neurons, encompassing an input layer, one or more hidden layers, and an output layer. Each neuron in the network receives input signals, undergoes transformation through an activation function, and transmits the result to neurons in the subsequent layer. Through an iterative process called backpropagation, the network adjusts the connection weights between neurons during training to minimize the error between predicted and actual outputs.

2.3 Support Vector Regression

Support Vector Regression (SVR) is an extension of support vector machines, where the decision function provides a numerical estimate for the dependent variable. It is a promising algorithm as it can overcome local minima and possesses generalization abilities and sparse representation [20]. According to Vapnik [20], the SVR algorithm is related to three concepts: (a) solutions obtained from an ideal hyperplane, (b) the complexity of dot products for non-linear solution surfaces, and (c) soft margins achieved through a set of slack variables [20].

2.4 Decision trees

A Decision Tree (DT) is a binary tree that represents a set of predictor variables used to predict the dependent variable. A DT consists of nodes, with the topmost node serving as the root and representing all rows in a dataset. Each node is split into two child nodes using a splitting variable [21]. The main advantage of a DT model is that it provides non-technical individuals with a comprehensive overview of an issue [22]. Additionally, it is a highly effective ML method for generating multiple potential outcomes before selecting the best model [23]. The accuracy of a DT depends on its complexity, which is determined by the depth of the tree, the number of nodes, leaves, and attributes, all of which are influenced by the stopping criteria and pruning technique employed [21].

3. Research Methodology

Figure 1 depicts the methodology of this research study. This section provides a detailed description of the dataset collection, the StackSDCE model, and the assessment metrics.

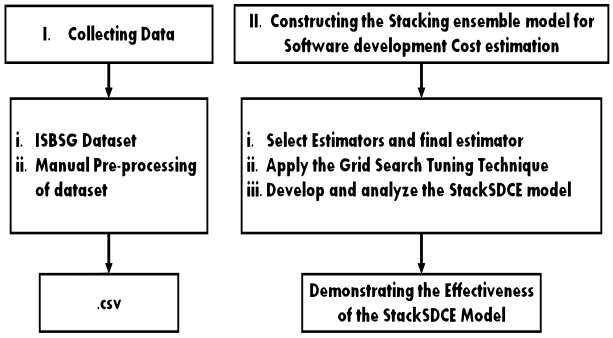

Figure 1. Research methodology.

3.1 Collecting Data

The ISBSG dataset contains comprehensive information on completed software projects, including practices, tools, and methodologies, as well as process and product data. This dataset serves multiple purposes such as benchmarking, monitoring, quality control, and performance management throughout the software development process. It is the largest accessible dataset for cost/effort-estimating research and has been extensively used in various publications. Given our focus on software development cost, we specifically chose data categorized under "new development" in the "development type" and using the "COSMIC FSM" method in the "count approach". Moreover, we exclusively consider data that are selected in our previous work [1]. Table 1 displays the data fields, the corresponding values selected for this study, the values that were discarded, and the number of projects. After the preprocessing phase, we opted for a total of seven attributes.

Table 1. First selection of data concerning software projects from the ISBSG dataset

ISBSG Data Field	**Selected Values**	**Discarded Values**	**Projects**
Count Approach	COSMIC	IFPUG, NESMA, FISMA, etc.	449
development Type	New development	Enhancement and Redev	374

The ISBSG projects underwent a thorough quality assurance process and preprocessing steps to refine the dataset for use in our cost estimation model. This involved several actions to ensure data integrity and homogeneity:

- a) We excluded attributes measured after project completion, as they were not relevant to our cost estimation model.
- b) Null records were identified for each attribute, and those with more than 40% null values were excluded.
- c) We considered the quality rating provided by ISBSG reviewers and excluded project information rated higher than grade 'B', prioritizing higher-quality data.
- d) Records containing null values in numerical attributes were removed to maintain data consistency.

After preprocessing, the filtered ISBSG dataset comprised 374 projects and 82 attributes. The selected attributes, along with their abbreviations and descriptions, are summarized in Table 2.

Table 2. ISBSG Variables.

ID	**CODE**	**Attribute Name**	**Description**
0	FCE	Full Cycle Work Effort	Total effort (in hours) recorded against the project.
1-4	CA	Count Approach	Description of the technique used to size the project.
5	AFP	Adjusted Function Points	Adjusted function point count adjusted by the Value Adjustment Factor.
6	PET	Project Elapsed Time	Total elapsed time for the project (in calendar months).
7	IY	Implementation Year	Actual year of implementation.
8-11	DTY	Development Type	Type of development: new development, enhancement, or re-development.
12-23	OT	Organisation Type	Type of organisation that submitted the project.
24-38	DT	Development Technique	Techniques used during development.
39-41	FS	Functional Sizing Technique	Technology to support the functional sizing process.
42-46	DP	Development Platform	Primary development platform.
47-52	LT	Language Type	Language type used for the project (e.g., 3GL, 4GL).
53-63	PPL	Primary Programming Language	Primary programming language used (e.g., Java, C++).
64-72	DBS	Database System	Primary technology database used.
73-79	RM	Recording Method	Method used to obtain effort data.
80	RL	Resource Level	People whose time is included in the work effort data reported.
81	MTS	Maximum Team Size	Maximum number of people that worked on the project.
82	ATS	Average Team Size	Average number of people that worked on the project.

3.2 Correlation Inference for Software Development Cost Estimation

González-Ladrón-de-Guevara et al. [24] analyzed SDCE using the ISBSG dataset. They conducted a review of various studies based on the ISBSG dataset from 2000 to 2013, aiming to identify the ISBSG variables commonly utilized in estimations and to assess the impact of these variables on the development of cost estimation models. The researchers identified both dependent and independent variables from a total of 71 ISBSG variables. Among these, 20 variables were frequently employed as independent variables in most studies. Their findings provide valuable insights for other researchers in the selection of appropriate variables for cost estimation models.

In this section, we use the ISBSG dataset to select the 17 attributes (numerical features) with 16 being independent variables and one serving as the dependent variable (Development Cost).

The Pearson correlation coefficient heat map is employed to analyze feature subsets (refer to Figure 2). The primary reason for selecting the Pearson correlation coefficient is its ability to assess all possible combinations of the specified features.

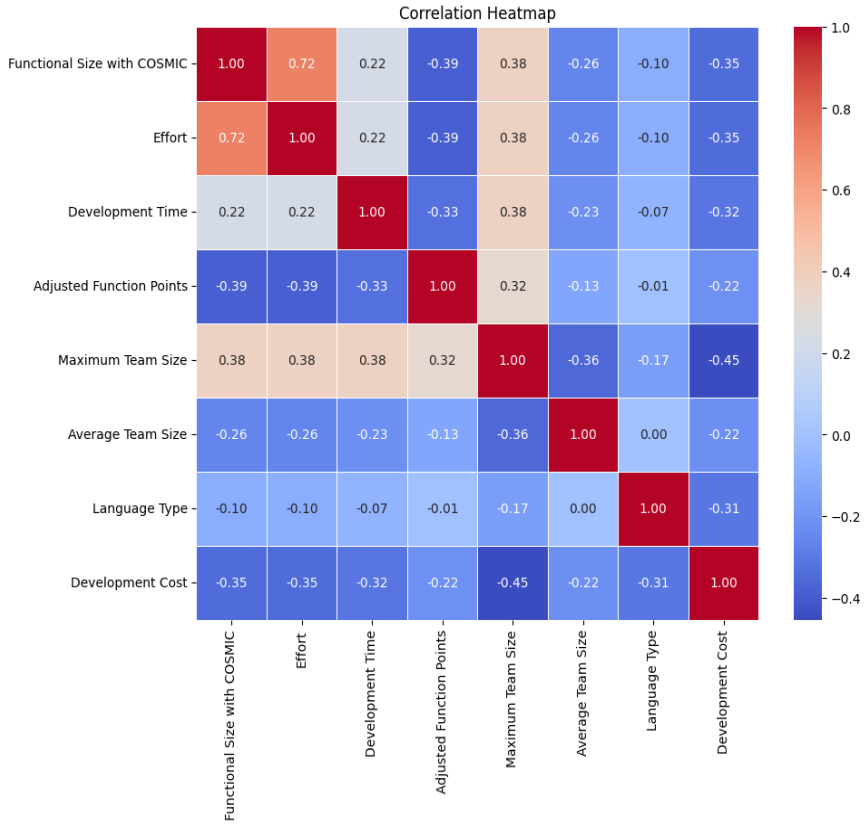

Figure 2. Pearson correlation heat map.

The provided correlation heatmap shows the correlation coefficients between the development cost and other variables in the dataset. Here's a brief analysis of the results:

- Functional Size with COSMIC: This variable has the highest correlation with the development cost among all the variables, with a correlation coefficient of 0.78. This indicates a strong positive correlation, suggesting that as the functional size with COSMIC increases, the development cost tends to increase as well.
- Effort: The effort also shows a strong positive correlation with the development cost, with a correlation coefficient of 0.78. This is expected, as higher effort is typically associated with higher costs in software development projects.
- Development Time: The correlation coefficient between development time and development cost is 0.65, indicating a moderate positive correlation. This suggests that longer development times tend to result in higher costs.
- Adjusted Function Points: This variable has a correlation coefficient of 0.53 with the development cost, indicating a moderate positive correlation. Adjusted function points are another measure of project size, and their correlation with development cost reinforces the relationship between project size and cost.
- Maximum Team Size and Average Team Size: Both variables show a moderate positive correlation with the development cost, with correlation coefficients of 0.72 and 0.57 respectively. This suggests that larger team sizes are associated with higher development costs.
- Language Type: The correlation coefficient between language type and development cost is 0.61, indicating a moderate positive correlation. Different programming languages may require different levels of expertise and resources, which can impact development costs. Overall, the heatmap provides valuable insights into the relationships between various project attributes and the development cost. It highlights the importance of considering factors such as project size, effort, development time, team size, and the choice of programming language when estimating development costs.

3.3 Constructing the StackSDCE Model

The SEM-building method is based on the idea that properly aggregating weak models can improve overall performance and accuracy. In our work, we examined two key factors to determine their effectiveness for developing SEMs. These parameters were defined in sci-kit-learn as follows (see Table 3): StackingRegressor(estimators, final_estimator=None).

Table 3. Parameters of the SEM.

Parameters	**Description**
Estimators	Base estimators that will be stacked together
Final_estimator	An estimator used to combine the base estimators

The "final estimator" parameter played a crucial role in determining the meta-model of our study. It was imperative to identify which models would function as the "estimators" and which would be designated as the "final estimator" before merging the predictions of the estimators into the selected final estimator to generate estimates.

Given the challenge of selecting an appropriate ML method with the requisite precision for constructing an estimation model, we opted to utilize NN, SVR, and DT to construct the StackSDCE model. These ML methods were chosen due to their widespread usage in software cost/effort estimation [19] and their varying levels of precision across different datasets.

3.3.1 Select estimators and final estimator

Construct models individually

This section discusses the hyperparameter values of the selected ML methods after employing GS tuning techniques. GS [13, 25] was utilized to fine-tune the three chosen ML methods and enhance their predictive accuracy by identifying the optimal configuration of hyperparameters through an exhaustive search. A 10-fold cross-validation method was employed to determine the best configuration in terms of Root Mean Square Error (RMSE) and Mean Absolute Error (MAE) using the error function. Table 4 lists the best configuration for each method that yields minimal MAE and RMSE.

Table 4. Parameters of the GS.

ML Technique	**Parameters**
Neural Network	"activation": "sigmoid", "batch-size": 32, "epochs": 20, "initializer": "normal", "loss": "mae", "optimizer": "rmsprop"
Support vector regression	"C": 1, "gamma": 0.01, "kernel": "linear"
Decision trees	"max-depth": 7, "max-features": "log2", "max-leaf-nodes": 10, "min-samples-leaf": 10, "min-weight-fraction-leaf": 0.01, "splitter": "best"

Results

Three evaluation criteria (RMSE, R^2, and MAE) were utilized to assess the overall performance of the selected prediction models. The MAE represents the average of the absolute differences between the actual and predicted cost [26]. It is calculated as follows:

$$\text{MAE} = \sum_{i=1}^{N} |x_i - y_i|$$

where y is the prediction, X is the true value, and N is the total number of projects.

The RMSE is the standard deviation or predicted errors of the residuals. It is calculated as follows:

$$\text{RMSE}(y, \hat{y}) = \sqrt{\frac{\sum_{i=0}^{N-1}(y_i - \hat{y}_i)^2}{N}}$$

where y is the prediction, $\hat{y}$ is the true value, and N is the total number of projects.

The coefficient of determination (R^2) regression score (sklearn.metrics.r2_score.html), evaluates how much the variance in the model, which is dependent on the independent variables (y), has been sufficiently explained. It is calculated as follows:

$$R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{(y_i - y)^2}$$

Table 5 presents the results of the assessment criteria. The highest R^2 score observed was 1.0. Figure 3 displays the accuracy results of the estimators' predictions. The DT exhibited the highest precision. Therefore, following the concept of the SEM, the most accurate StackSDCE prediction could be achieved by combining the weak estimators, SVR and NN, and utilizing the DT as the final estimator.

Table 5. Prediction analysis using the R^2 score.

Method/Parameters	**R^2 Score**
Neural Network	0.4
Support Vector Regression	0.3
Decision Tree Regression	0.57

Figure 3 illustrates the (R^2 score of the selected ML "estimators". As observed, the DT model outperformed the NN model and the SVR model.

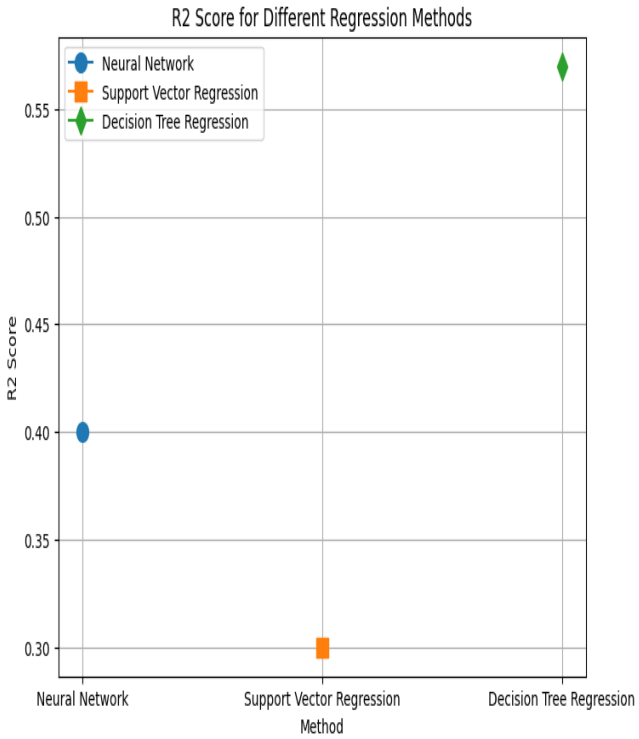

Figure 3. R^2 score.

As depicted in Table 6, error metrics provide valuable insights into the performance of the DT model. After tuning with GS, the DT model achieved an MAE of 0.0712 and an RMSE of 0.1062. Notably, the optimal MAE and

RMSE scores are both “0”. These metrics offer detailed information regarding the accuracy and precision of the DT model’s predictions, indicating its effectiveness in estimating software development costs.

Figure 4 presents a comprehensive overview of the error results obtained from the estimators’ predictions. The visualization provides detailed insights into the performance of each model. Notably, the DT model stands out for its superior precision compared to other estimators. This observation underscores the significance of employing a SEM approach, wherein the strengths of individual models are leveraged to enhance overall prediction accuracy. By combining the weaker estimators, such as SVR and NN, with the robust performance of the DT model as the “final estimator”, the StackSDCE model can achieve heightened accuracy and reliability in SDCE.

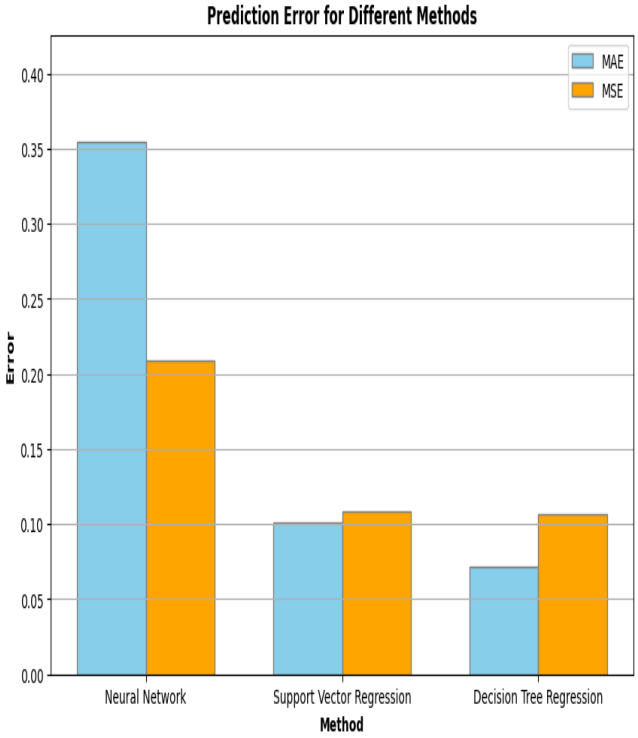

Figure 4. Prediction Errors for different methods.

Table 6. Prediction evaluation using MAE and RMSE.

Method/Parameters	**MAE**	**RMSE**
NN	0.3548	0.2092
SVR	0.1012	0.1081
DT	0.0712	0.1062

3.3.2 Development and analysis of the StackSDCE model

Based on the insights derived from the detailed examination illustrated in Figure 4, the DT model emerged as the optimal choice for the final estimator. Following a rigorous tuning process facilitated by GS, the parameters of the SEM were meticulously refined, as outlined in Table 7.

Table 7. Parameters of the SEM with GS.

Name of ML Techniques	**The Parameters**
Stacking model	estimators=[(Neural Network, SVR)], final estimator= Decision Tree Regression()
DT	"max-depth": 4, "max-features": "log2", "max-leaf-nodes": 10, "min-samples-leaf": 10, "min-weight-fraction-leaf": 0.01, "splitter": "best"

Table 8 and Figure 5 provide detailed insights into the updated R^2 score for all three individual ML methods and compare them to the SEM. The results indicate that the new StackSDCE model, configured with GS parameters, exhibited enhanced performance compared to the initial StackSDCE model constructed without GS tuning. This suggests that the application of GS tuning effectively optimized the model's parameters, resulting in improved accuracy and predictive capability in SDCE.

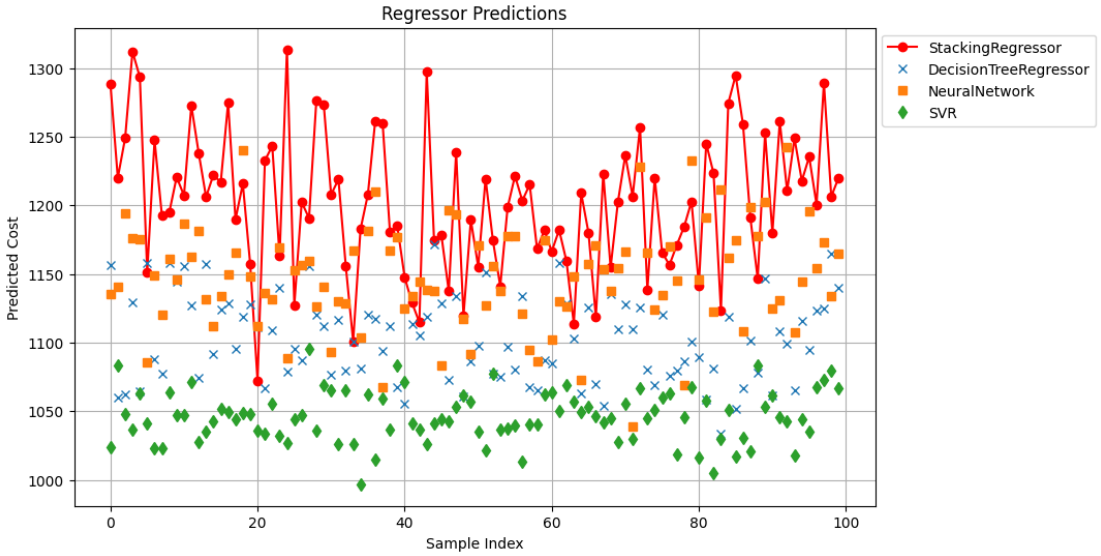

Figure 5. Regressor estimation score.

Table 8. Prediction evaluation using the R^2 score.

Method/Parameters	**R^2 Score**
Neural Network	0.54
Support Vector Regression	0.61
Decision Tree Regression	0.72
Stacking Regressor	0.89

Figure 6 provides a visual representation of the ML "estimators" alongside the average of their predictions. Upon observation, it is clear that the StackSDCE model exhibits superior performance compared to the individual NN, SVR, and DT models. This suggests that the ensemble approach effectively leverages the strengths of multiple models to enhance overall prediction accuracy in SDCE.

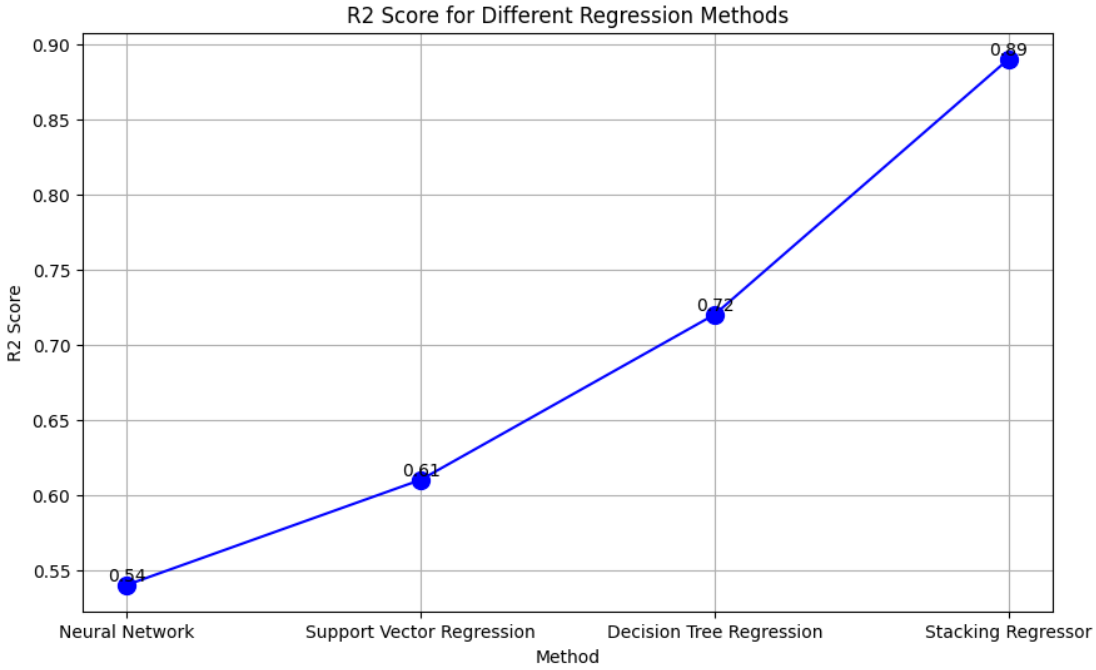

Figure 6. Regressor estimation score.

4. Discussion and Comparaison

This study delved into three key challenges aimed at enhancing the accuracy of the SDCE model. Initially, the ISBSG dataset served as the foundation for training and testing three distinct ML techniques: NN, SVR, and DT, with the functional change request size serving as the primary independent variable.

A Pearson correlation analysis is conducted to discern the relationships among the various features pertinent to cost development estimation. Subsequently, GS tuning was employed to meticulously optimize the hyperparameters of the selected ML techniques. Finally, an SEM was constructed by combining the diverse ML methods, thereby harnessing their collective strengths for improved predictive performance. Therefore, the functional size of a development request, as determined by the Functional Size Measurement (FSM) of the International Function Point Users Group (IFPUG), holds significance as it enhances the precision of the SDCE model. Sakhrawi et al. [1] reported that, unlike in the field of software enhancement effort estimation, such as total effort estimation, the Functional Size Measurement (FSM) of the Common Software Measurement International Consortium (COSMIC) performs better than the FSM of IFPUG.

As illustrated in Figure 1, the construction of the StackSDCE model entails two main stages: (i) selection of the estimators and the final estimator, and (ii) building the StackSDCE model. During the construction of the first StackSDCE model, a series of tests were conducted using the three chosen ML methods by iteratively adjusting their parameter values. These tests revealed that the DT model produced the most precise outcomes, hence it was utilized as the final estimator. In the construction of the second StackSDCE model, GS was employed to fine-tune the parameters of the chosen ML methods. Once again, the DT model was selected as the final estimator.

The findings of the constructed models are summarized as follows:

- The StackSDCE model, comprising the NN, SVR, and DT models, produced more accurate results than the three models individually. Employing GS to fine-tune and identify optimal hyperparameters significantly improves the precision of the StackSDCE model.
- Using FS as a primary independent variable demonstrated superior performance, indicating that utilizing GS tuning to configure the ML models yields favorable outcomes.

Test results reveal the following insights:

- The Decision Tree (DT) model demonstrates the highest precision, boasting a smaller Mean Absolute Error (MAE) of 0.07 and a higher R-squared (R^2) score of 0.7 in comparison to the SVR and NN models.
- Employing GS to fine-tune and configure the selected ML methods significantly enhances the precision of the StackSDCE model, resulting in an impressive R-squared (R^2) score of 0.8. This performance surpasses the precision achieved through manual configuration of the chosen ML methods.

These detailed findings underscore the effectiveness of utilizing an SEM approach, highlighting the importance of model selection and parameter tuning in optimizing SDCE accuracy.

5. Threats to Validity

While this study employed various evaluation methods for experimental comparison, its scope remains limited, posing threats to both internal and external validity.

Internal Validity:

Internal validity is primarily concerned with the evaluation method used to assess the proposed Software Development Cost Estimation (SDCE) model. Limitations in this regard include the size of the dataset, where a larger number of instances could enhance validity. Additionally, the number and nature of attributes utilized to construct the StackSDCE model may impact internal validity. Employing feature selection methods could mitigate this limitation [16, 27].

External Validity:

External validity relates to the generalizability of the study's results. One threat arises from the exclusive use of numerical data, potentially limiting the findings. Incorporating both numerical and categorical data types could offer a more comprehensive understanding of the model's performance. Another threat pertains to the dataset choice, as only the ISBSG dataset was utilized. To bolster external validity, future research should explore alternative datasets. Additionally, the reliance on a single tuning method, GS, poses a threat to external validity. Incorporating other tuning methods, such as Genetic Algorithms (GAs), could enhance the generalizability of the study's findings.

6. Conclusions

Accurate software development cost estimates are crucial for effective project management, as they reduce uncertainty and contribute to project success. In contrast, inaccurate estimates can lead to customer dissatisfaction and project failure.

The empirical findings of this study are summarized as follows:

- The SEM enhances the accuracy of SDCE.
- GS tuning proves to be an effective technique for improving the accuracy of both individual models and stacked models. It facilitates the rapid and accurate selection of optimal parameter values.

Several key factors influence the accuracy of SDCE:

- Selection of a new dataset: Recognizing the impact of technology used in software development, maintenance, and testing on the required cost/effort.
- Choice of independent variable: Functional size emerges as a critical factor in improving the accuracy of SDCE-based ensemble learning, as demonstrated by multiple studies, including this one.
- Configuration of ML method parameters: GS tuning proves to be an efficient method for quickly and effectively configuring model parameters.

This study aims to further enhance the accuracy of SDCE by employing more effective ML methods in the future. Additionally, it strives to develop a decision-making tool that automates the process of SDCE.

Author Contributions

Conceived and designed the analysis (all manuscript's authors); collected the data (Z.S.); contributed data and analysis tools (Z.S.); performed the analysis (all manuscript's authors); and wrote the paper (all manuscript's authors). All authors have read and agreed to the published version of the manuscript.

Funding

The authors declare that they have no funding for the present study.

Conflict of Interest Statement

The authors declare that they have no conflicts of interest to report regarding the present study.

Data Availability Statement

ISBSG dataset, available in (https://www.isbsg.org/project-data/).

References

[1] Z. Sakhrawi, T. Labidi, A. Sellami, N. Bouassida. Data Quality Improvement for More Accurate Regression Test Effort Estimation. *In Intelligent Systems Design and Applications: 19th International Conference on Intelligent Systems Design and Applications (ISDA 2023)* held December 11-13, 2023, Volume 6, Springer International Publishing.

[2] M. Maher, JS. Alneamy. An Ensemble Model for Software Development Cost Estimation. *In 2022 5th International Seminar on Research of Information Technology and Intelligent Systems (ISRITI)* pp. 346-350. IEEE. 2022.

[3] J. Singh, K.S. Dhindsa, J.Singh. Reengineering cost estimation using scrum agile methodology. *International Journal of Computer Information Systems and Industrial Management Applications*, 11,pp.11-11. 2019.

[4] A. Syed Sarmad, J. Ren, K. Zhang, J. Wu, C. Liu. Heterogeneous ensemble model to optimize software effort estimation accuracy. *IEEE Access*. 13;11:27759-92. 2023.

[5] LL. Minku, X. Yao. A principled evaluation of ensembles of learning machines for software effort estimation. *In Proceedings of the 7th international conference on predictive models in software engineering*, pp. 1-10. 2011.

[6] J. Wen, S. Li, Z. Lin, Y. Hu, 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.

[7] C. Lopez-Martin. Machine learning techniques for software testing effort prediction, *Software Quality Journal*, vol. 30, no.1, pp. 65–100, 2022.

[8] AG. Priya Varshini, K. Anitha Kumari, V. Varadarajan. Estimating software development efforts using a random forest-based stacked ensemble approach, *Electronics*, vol. 10, no.10, pp. 1–19, 2021.

[9] Z. Sakhrawi, A. Sellami, N. Bouassida. Support vector regression for enhancement effort prediction of Scrum projects from COSMIC functional size, *Innovations in Systems and Software Engineering*, vol. 18, no.1, pp. 137–153, 2022.

[10] Z. Sakhrawi, A. Sellami, N. Bouassida. Software Enhancement Effort Estimation using Machine Learning Regression Methods, *International Journal of Computer Information Systems and Industrial Management Applications*, vol. 12, pp. 412–423, 2020.

[11] SS. Ali, MS. Zafar, MT. Saeed. Effort Estimation Problems in Software Maintenance–A Survey,*3rd International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)*, pp.1–9, 2020.

[12] A. Idri, M. Hosni, A. Abran. Systematic literature review of ensemble effort estimation, *Journal of Systems and Software*, vol. 118, pp. 151–175, 2016.

[13] T. Labidi, Z. Sakhrawi. On the value of parameter tuning in stacking ensemble model for software regression test effort estimation. *The Journal of Supercomputing*. 79(15):17123-17145, 2023.

[14] DK. NC, KV. Suresh. Ensemble Learning for Static Hand Gesture Recognition using HOG and LBP Features on RGB-D Data. *International Journal of Computer Information Systems and Industrial Management Applications.* Jan 1;15,2023

[15] P. Sampath Kumar, R. Venkatesan. Improving Accuracy of Software Estimation Using Stacking Ensemble Method, in. *Advances in Machine Learning and Computational Intelligence*, Springer, pp. 219–227, 2021.

[16] Z. Sakhrawi, A. Sellami, N. Bouassida. Software enhancement effort estimation using correlation-based feature selection and stacking ensemble method, *Cluster Computing*, vol. 25, no.4, pp. 2779–2792, 2021.

[17] Z. Sakhrawi, A. Sellami, N. Bouassida. Software Enhancement Effort Estimation using Stacking Ensemble Model within the Scrum Projects: A Proposed Web Interface, in. *Proceedings of the 17th International Conference on Software Technologies ICSOFT*, Lisbon, Portugal, pp. 91–100, 2022.

[18] M. Hosni, A. Idri, A. Abran, A. Bou Nassif. On the value of parameter tuning in heterogeneous ensembles effort estimation, *Soft Computing*, vol. 22, no.18, pp. 5977–6010, 2018.

[19] PS. Kumar, HS. Behera, A. Kumari, J. Nayak, B. Naik. Advancement from neural networks to deep learning in software effort estimation: Perspective of two decades, *Computer Science Review* 38, 100288, 2020.

[20] C. Cortes, V. Vapnik. Support-vector networks, in. *Machine Learning*, Springer, vol. 20, no.3, pp. 273–297, 1995.

[21] M. Kantardzic. Data mining: concepts, models, methods, and algorithms. *Wiley-IEEE Press*, 2011.

[22] AB. Nassif, M. Azzeh, LF. Capretz, D. Ho. A comparison between decision trees and decision tree forest models for software development effort estimation, *In 2013 Third International Conference on Communications and Information Technology (ICCIT)*, pp. 220-224. IEEE, 2013.

[23] S. Mehta, KS. Patnaik. Improved prediction of software defects using ensemble machine learning techniques,

Neural Computing and Applications 33: 10551-10562, 2021.

[24] F. González-Ladrón-de-Guevara, M. Fernández-Diego, C. Lokan. The usage of ISBSG data fields in software effort estimation: A systematic mapping study. *Journal of Systems and Software*, 113, p.188-215, 2016.

[25] X. Ma, Y. Zhang, Y. Wang. Performance evaluation of kernel functions based on grid search for support vector regression, in. *7th international conference on cybernetics and intelligent systems (CIS) and IEEE conference on robotics,* automation and mechatronics (RAM), Cambodia, pp. 283–288, 2015.

[26] M. Shepperd, S. MacDonell. Evaluating prediction systems in software project estimation, *Information and Software Technology*, 54, no. 8, pp. 820-827, 2012.

[27] T. Labidi, Z. Sakhrawi, A. Sellami, A. Mtibaa, N. Bouassida. On the use of OLS regression algorithm and Pearson correlation algorithm for improving the SLA establishment process in cloud computing, *Innovations in Systems and Software Engineering*, vol. 18, no.1, pp. 215-229, 2022.

Author Biographies

Zaineb Sakhrawi. is a Ph.D. in computer science with a specialization in Software Engineering. She has four years of experience in teaching. Her research interests include software maintenance, software testing, software quality, effort estimation, software measurement, ontology, and Machine Learning techniques. She works as a contractual assistant at the Higher Institute of Computer Science and Multimedia. She is a Multimedia, Information Systems, and Advanced Computing Laboratory (MIRACL) member.

Taher Labidi. received a Ph.D. in Computer and Information Science from the University of Sfax, Tunisia. Previously, he served as an Automotive Software Test and Validation Engineer at Technica Engineering, Tunisia, holding ISTQB® certification. Currently, he holds the position of Assistant Professor at the Department of Computer Science, Higher Institute of Computer Science of Medenine, University of Gabes, Tunisia. He is a member of the Multimedia, Information Systems, and Advanced Computing Laboratory at the University of Sfax. His diverse research interests encompass Semantic Web and Ontology, Software Testing, Cloud Computing, Machine Learning, and Medical Information Systems.

Asma Sellami is teaching at the University of Sfax in Tunisia. Her current research interest includes broadly measurement in Software Engineering, software quality and software project management. She is also working on ISO standards for measuring the functional size of software, and has been involved in developing case study of ISO 19761 (COSMIC FSM Method). She published more than 40 referred conferences, journals, and technical reports. She is currently member of COSMIC Advisory council in Tunisia.

Nadia Bouassida received a Phd in Computer and Information Science from the University of Science of Tunis, Tunisia. Currently, she is Associate Professor at the Department of Computer Science of the Higher Institute of Computer Science and Multimedia at the University of Sfax, Tunisia. She is a member of the Multimedia, Information systems and Advanced Computing Laboratory, University of Sfax. Her research interests include reuse techniques, such as design patterns, Frameworks and Software Product Lines.