



# Machine Learning for Accurate Software Development Cost Estimation in Economically and Technically Limited Environments

Mohammad Alauthman, Department of Information Security, Faculty of Information Technology, University of Petra, Amman, Jordan

 <https://orcid.org/0000-0003-0319-1968>

Ahmad al-Qerem, Computer Science Department, Faculty of Information Technology, Zarqa University, Jordan

Someah Alangari, Department of Computer Science, College of Science and Humanities, Shaqra University, Saudi Arabia

 <https://orcid.org/0000-0003-1308-1762>

Ali Mohd Ali, Communications and Computer Engineering Department, Faculty of Engineering, Al-Ahliyya Amman University, Jordan

Ahmad Nabo, Software Engineering Department, Faculty of Information Technology, Zarqa University, Jordan

Amjad Aldweesh, College of Computing and IT, Shaqra University, Saudi Arabia\*

Issam Jebreen, Computer Science Department, Faculty of Information Technology, Zarqa University, Jordan

Ammar Almomani, Skyline University College, UAE

Brij B. Gupta, CCRI, Department of Computer Science and Information Engineering, Asia University, Taichung, Taiwan & Symbiosis Centre for Information Technology (SCIT), Symbiosis International University, Pune, India & School of Computing, Skyline University, UAE, & Lebanese American University, Beirut, Lebanon & Center for Interdisciplinary Research, University of Petroleum and Energy Studies (UPES), Dehradun, India

## ABSTRACT

Cost estimation for software development is crucial for project planning and management. Several regression models have been developed to predict software development costs, using historical datasets of previous projects. Accurate cost estimation in software development is heavily influenced by the relevance and quality of the cost estimation dataset and its suitability to the software development environment. The currently available cost estimation datasets are limited to North American and European environments, leaving a gap in the representation of other economically and technically constrained software industries. In this article, the authors evaluate the performance of regression models using the SEERA dataset, which highly represents these constrained environments. This study provides insights into selecting regression models for cost estimation in software development. It highlights the importance of using appropriate models based on the specific software development model and dataset used in the estimation process. In the performance evaluations of eight regression models, including elastic net, lasso regression, linear regression, neural network, RANSACRegressor, random forest, ridge regression, and SVM, for cost estimation in different software models, along with correlation coefficients and accuracy indicators, were reported. The results showed that SVM and random forest indicated superior performance. However, the elastic net, lasso regression, linear regression, neural network, and RANSACRegressor models also demonstrated exemplary performance in cost estimation.

## KEYWORDS

Constrained Environments, Cost Estimation, Regression Models, SEERA Dataset, Software Development

DOI: 10.4018/IJSSCI.331753

\*Corresponding Author

This article published as an Open Access article distributed under the terms of the Creative Commons Attribution License (<http://creativecommons.org/licenses/by/4.0/>) which permits unrestricted use, distribution, and production in any medium, provided the author of the original work and original publication source are properly credited.

## 1. INTRODUCTION

Cost estimation is a critical aspect of software development (Rankovic, Rankovic, Ivanovic, & Lazic, 2021; Rankovic, Rankovic, Ivanovic, & Lazic, 2021; Mukherjee & Malu, 2014), as it helps in predicting the resources required for the project and ensuring that the project is completed within budget and on time (Pandey et al., 2020). However, estimating the cost and effort for different software development models can be challenging due to their unique characteristics and requirements (Boehm, 2017; Kumar et al., 2020).

Several essential features must be considered when estimating the cost and effort for different software development models. One of the most crucial factors is the size of the project, which refers to the number of software components or functions that need to be developed. The larger the project, the more effort and resources it will require, ultimately impacting the cost estimation (Saavedra Martínez et al., 2020; Mahmood et al., 2021). The project's complexity is another critical feature affecting cost and effort estimation. The complexity of the software model can vary based on various factors, such as the number of interrelated components, the number of decision points, and the level of customization required. Developing more complex software models will require more effort and resources, resulting in higher costs (Mahmood et al., 2021).

The development team's expertise is another vital factor when estimating the cost and effort for different software models. The level of experience, knowledge, and skills of the team will significantly impact the development time and cost. A team with more experience and knowledge can develop a project more efficiently, resulting in lower costs (Nassif et al., 2019).

The development process also plays a crucial role in cost and effort estimation. The development process can be iterative or sequential, and each approach has advantages and disadvantages. The sequential approach, also known as the Waterfall model, is more structured, which can help ensure that each development phase is completed before moving on to the next. In contrast, the iterative approach, the Agile model, is more flexible and adaptable, allowing for changes throughout the development process.

Finally, the software development environment also affects cost and effort estimation. The environment can include hardware and software tools, such as integrated development environments, version control systems, and testing tools necessary to complete the project. The cost and availability of these tools and resources will impact the cost estimation for the project. As a result, several important features need to be considered when estimating the cost and effort for different software models. These include the project's size and complexity, the development team's expertise, the development process, and the software development environment. An accurate model for estimating the cost and effort will ensure the project is completed within budget and on time, providing significant benefits to the development team and the organization.

We observed first-hand the challenges that local software development teams faced due to limited resources and infrastructure constraints. Accurately estimating costs was critical for project planning and management under these conditions. However, existing cost estimation techniques and datasets did not adequately account for the realities of working in such constrained environments. We were motivated to address this research gap and help improve cost estimation practices for software teams operating under similar limitations.

Machine learning plays an essential role in determining the critical features that affect the cost and effort estimation of different software models (Safari & Erfani, 2020; Holtkamp et al., 2015; Casado-Lumbreras et al., 2014). With the help of machine learning algorithms, large and complex datasets can be analyzed to identify patterns and relationships between cost and effort variables and the factors that affect them. By utilizing machine learning techniques, such as regression analysis, decision trees, and neural networks, it is possible to identify the most significant variables that impact software development cost and effort.

The ability of machine learning algorithms to identify essential features can lead to more accurate and reliable cost and effort estimates for different software development models. Machine learning algorithms can produce more accurate models considering the complex nature of software development projects by considering a wide range of features, including technical, organizational, and cultural factors. Furthermore, machine learning techniques can continually update and refine cost and effort estimation models as new data becomes available, ensuring that the models remain accurate and current.

The key gaps we identified in previous research were the lack of focus on non-Western software development contexts and the inadequacy of existing cost estimation datasets in representing technically and economically constrained industries. Most available datasets represented North American and European environments, leaving significant room to improve the external validity and generalizability of cost estimation models for other global contexts.

Overall, using machine learning algorithms to determine the critical features that affect the cost and effort estimation of different software models is essential in providing accurate and reliable estimates. With the ever-increasing complexity of software development projects, identifying and considering a wide range of factors is crucial in ensuring that projects are delivered on time and within budget. Machine learning provides a powerful tool to help achieve these goals, enabling more accurate and reliable cost and effort estimation for software development projects of all types and sizes. The rest of this paper is organized as follows: In Section 2, we provide the related works. Section 3 presents the methodology and data used in this study. In Section 4, we present the results and discussion. Finally, Section 5 provides conclusions and suggestions for future work.

## 2. RELATED WORKS

Cost estimation for software development varies, depending on the development model and the application domains. Each application domain has its unique characteristics, requirements, and constraints, which may affect the cost of development (Ilyas et al., 2020; Akbar et al., 2019). The cost estimation model for an application domain must consider factors, such as the system's complexity, size, functionality, and technologies. For instance, a software system for a financial application may require complex algorithms for data processing, security features, and integration with multiple databases. In contrast, a software system for a simple game may require less complex functionalities. The choice of the development model also affects the cost estimation. For instance, the waterfall model may suit application domains requiring a well-defined and predictable process.

In contrast, the agile model may suit application domains requiring frequent changes and iterations. In conclusion, the cost estimation model for software development must consider both the application domain and the development model to produce accurate and reliable cost estimation (Ali & Gravino, 2021).

There are various approaches for cost estimation in software development, each with its strengths and weaknesses. One of the most common approaches is the algorithmic model, which uses mathematical formulas to estimate the cost of software development based on project size and complexity. This approach works well for well-defined projects, with general requirements and specifications. However, it may not be suitable for more complex or innovative projects, without fully defined requirements. Another approach is the expert judgment model, which relies on the expertise of experienced professionals to estimate the cost of software development. This approach can be practical for complex and innovative projects, allowing for greater flexibility and adaptability in the estimation process. However, it is highly subjective and may be affected by biases or limitations in the knowledge and experience of the experts involved. Machine learning algorithms have also been used for cost estimation in software development, with promising results (Kumar et al., 2021; Zhao & Zhang, 2020; Panda & Majhi, 2020; Promise Software Engineering Repository, n.d.). These algorithms can analyze large amounts of data and identify patterns and relationships that are not

**Table 1. Comparing different software development models**

Software Development Model	Description	Advantages	Disadvantages	Software Development Model	Description
WATERFALL MODEL	A linear sequential approach where each phase must be completed before moving on to the next phase.	Clear and structured development process.	Not suitable for projects with changing requirements.	Waterfall Model	A linear sequential approach where each phase must be completed before moving on to the next phase.
AGILE MODEL	An iterative and incremental approach with a focus on adaptability and customer satisfaction.	Flexibility to adapt to changing requirements. Collaboration between developers and customers.	Requires active customer involvement and can be challenging to manage for large projects.	Agile Model	An iterative and incremental approach with a focus on adaptability and customer satisfaction.
SPIRAL MODEL	A risk-driven model that involves continuous feedback loops and risk analysis.	Allows for better risk management and more accurate cost and schedule estimation.	Complex and can be time-consuming.	Spiral Model	A risk-driven model that involves continuous feedback loops and risk analysis.
V-MODEL	A sequential approach with related testing activities for each development stage.	Ensures that all requirements are met and that testing is integrated throughout development.	It can be inflexible and may not allow for changes in requirements.	V-Model	A sequential approach with related testing activities for each development stage.
INCREMENTAL MODEL	A model where software is developed in smaller increments or modules.	Allows for faster delivery and testing of working software.	It can be challenging to manage larger projects and may require more resources.	Incremental Model	A model where software is developed in smaller increments or modules.

easily discernible through other approaches. They can also adapt to changing project conditions and provide more accurate and reliable estimates. Table 2 compares different machine learning algorithms used for cost estimation in software development. The most effective approach for cost estimation in software development will depend on the specific characteristics of the project and the available resources and expertise. A combination of different approaches, including algorithmic models, expert judgment, and machine learning, may be needed to achieve the most accurate and reliable estimates (Chhabra & Singh, 2020; Mohammed & Jamal, 2022).

Comparing mathematical formulas and machine learning regression for cost estimation in different software models is a complex task that requires a thorough analysis of the strengths and limitations of each approach (Jha & Jha, 2020; Emtinan, 2020). Here are some potential points of Comparison:

**Table 2. Comparison between different machine learning algorithms used for cost estimation in software development**

Software Development Model	Dataset Name	Machine Learning Model	Performance Measures	Reference
WATERFALL	COCOMO	Neural Network	RMSE: 4.8, R2: 0.87	(Popović & Bojić, 2012)
WATERFALL	Desharnais	Random Forest	RMSE: 6.1, R2: 0.72	(Rankovic, Rankovic, Ivanovic, & Lasic, 2021)
AGILE	ISBSG	SVM	RMSE: 2.1, R2: 0.95	(Sakhrawi et al., 2020)
AGILE	SEERA	Lasso Regression	RMSE: 2.4, R2: 0.91	(Salmanoglu et al., 2017)
HYBRID	Albrecht	Linear Regression	RMSE: 3.7, R2: 0.81	(Di Martino et al., 2020)
HYBRID	NASA-TLX	Gradient Boosting	RMSE: 2.3, R2: 0.94	(Ali & Gravino, 2019)

Note: RMSE stands for Root Mean Square Error, and R2 stands for Coefficient of Determination. These performance measures are commonly used in evaluating the accuracy of machine learning models for cost estimation.

**Accuracy:** Machine learning regression algorithms have the potential to offer more accurate cost estimation than mathematical formulas. It can consider more variables and nonlinear relationships between variables. However, the accuracy of machine learning models depends on the quality and diversity of the training data, as well as the choice of algorithm and hyperparameters. Mathematical formulas can be accurate for simple software models, but may not be able to capture the complexity of more advanced models.

**Interpretability:** Mathematical formulas are often more interpretable than machine learning models. It provides explicit equations that can be used to calculate cost estimates based on input variables. Machine learning models, on the other hand, are often seen as black boxes because it can be challenging to understand how they arrive at their predictions. However, some machine learning models, such as decision trees, can provide interpretable rules that can be used to explain their predictions.

**Flexibility:** It can adapt to new data and software models without requiring manual adjustments. In contrast, mathematical formulas can become obsolete if the underlying software model changes or new variables are necessary for cost estimation.

**Data requirements:** Machine learning regression algorithms generally require more data than mathematical formulas to achieve reasonable accuracy. It needs to learn patterns in the data, which requires a large and diverse dataset. Mathematical formulas, on the other hand, can often be derived from a smaller dataset or expert knowledge.

**Development and maintenance:** Developing and maintaining a machine learning regression model can be more time-consuming and resource-intensive than creating a mathematical formula. It requires machine learning algorithms, data preprocessing, feature selection, and hyperparameter tuning expertise. In contrast, mathematical formulas can be developed and maintained by domain experts, without requiring specialized knowledge in machine learning.

Different mathematical formulas can be used for cost estimation in different software models. Here are a few examples:

**COCOMO (Constructive Cost Model):** This is a popular model for estimating software project cost, which uses a set of equations based on project size, complexity, and development environment. The COCOMO model has three versions: Basic, Intermediate, and Advanced.

**Function Points Analysis:** This model uses the number of function points in a software system to estimate development effort and cost. Function points are a measure of the functionality provided by the system and can be calculated based on the number of inputs, outputs, inquiries, files, and interfaces in the system.

**Putnam Model:** This model uses a set of equations based on project size, development team experience, and development environment to estimate project cost and schedule. The model also considers the number of software defects likely to be found during development and testing.

**PERT (Program Evaluation and Review Technique):** This model uses a probabilistic approach to estimate project costs and schedules. PERT uses three different estimates for each activity in the project: optimistic, most likely, and pessimistic. These estimates are then used to calculate a weighted average for each activity, which is used to estimate the overall project cost and schedule.

**Function Point Analysis Mark II:** This original model extension includes additional factors, such as data communication, distributed data processing, and transaction rates.

These formulas are often used with other techniques, such as expert judgment and historical data analysis, and may not always provide accurate estimates. Machine learning regression models can also supplement or replace these formulas in some cases, mainly when dealing with complex and diverse datasets. Table 3 compares the traditional and machine learning approaches.

The General Framework for software cost estimation prediction, using machine learning regression and preprocessing, can be applied to SEERA, a software cost estimation dataset for constrained environments. SEERA is a publicly available dataset containing data on software development projects subject to time and cost constraints (Panda & Majhi, 2020). The dataset can be used to develop

**Table 3. Comparing mathematical formulas and machine learning regression for cost estimation in different software models**

Comparison Criteria	Mathematical Formulas	Machine Learning Regression
ACCURACY	Accurate for simple models, it may not capture the complexity of advanced models.	Potential for higher accuracy due to the ability to consider more variables and nonlinear relationships. Dependent on quality and diversity of training data and algorithm/hyperparameters choice.
INTERPRETABILITY	More interpretable due to explicit equations for cost estimates.	Some models, such as decision trees, often seen as black boxes, can provide interpretable prediction rules.
FLEXIBILITY	It can become obsolete with changes to software models or new essential variables.	More adaptable to new data and software models without manual adjustments.
DATA REQUIREMENTS	It can often be derived from smaller datasets or expert knowledge.	Require large and diverse datasets to learn patterns and achieve good accuracy.
DEVELOPMENT AND MAINTENANCE	Can be developed and maintained by domain experts without specialized knowledge in machine learning.	More time-consuming and resource-intensive due to the expertise required in machine learning algorithms, data preprocessing, feature selection, and hyperparameter tuning.

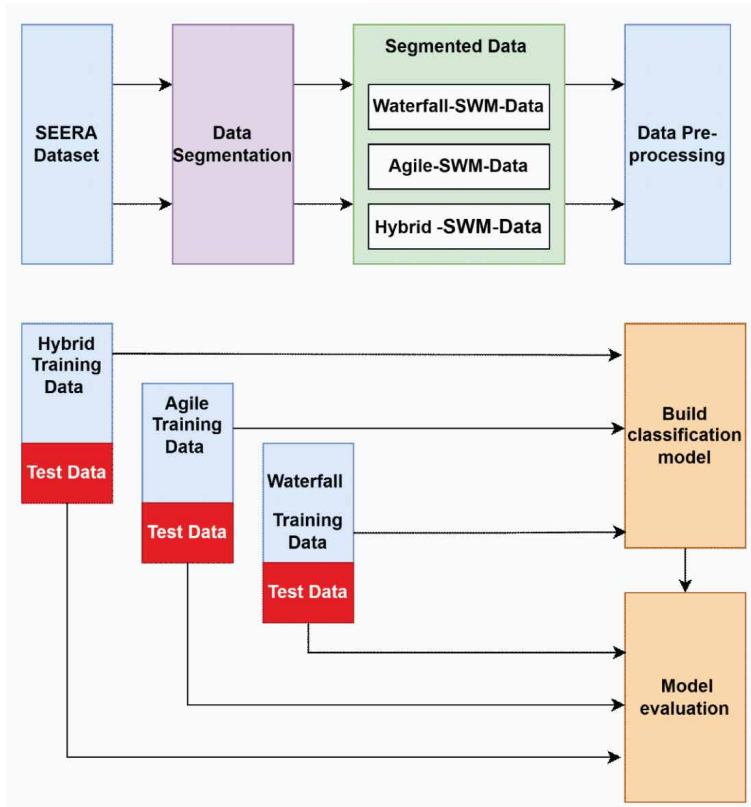
and evaluate machine learning models for predicting software development costs in constrained environments. The first step in preprocessing the SEERA dataset is to clean the data and remove any irrelevant or redundant data. Missing values are then filled, and any necessary feature engineering is applied. Feature engineering in the SEERA dataset includes using expert judgment, process metrics, and other software metrics to create new features that can provide valuable insights into the software development process. The SEERA dataset (Al Asheeri & Hammad, 2019), a software cost estimation dataset for constrained environments, can be segmented based on Waterfall, Agile, and Hybrid software development models. The Waterfall model’s development process is sequential, and each project phase must be completed before moving on to the next one. The SEERA dataset can be segmented using the Waterfall model. The data can be used to train machine learning models to accurately predict the costs of software development projects that follow the Waterfall model. In the Agile model, the development process is iterative and incremental, and the SEERA dataset can be segmented based on the use of the Agile model. Machine learning models can be trained on the data to accurately predict the costs of software development projects that follow the agile model. A combination of the Waterfall and Agile models is used in the Hybrid model, and the SEERA dataset can be segmented based on the Hybrid model. The data can be used to train machine learning models to accurately predict the costs of software development projects that follow the Hybrid model.

Segmenting the SEERA dataset based on software development models allows for more accurate predictions of software development costs for specific software development projects. Machine learning models can be trained on the segmented data to accurately predict the costs of software development projects that follow a particular development model, helping organizations to make informed decisions regarding software development budgets and timelines.

After feature engineering, the data is normalized to ensure that features are on the same scale and do not bias the model’s performance. Normalization helps ensure that features with large ranges do not overshadow smaller ranges.

The final step in preprocessing the SEERA dataset segmented based on S.W. development models is splitting the data into training and testing sets. The training set is used to train the machine learning model, and the testing set is used to evaluate its performance. Fig . 1 shows the SW-cost estimation prediction framework, taking care of the data preprocessing phase. The framework helps organizations make informed decisions regarding software development budgets and timelines in constrained environments.

Figure 1. General framework for SW-cost estimation prediction



### 3. MATERIALS AND METHODS

#### 3.1 Experimental Data

This paper uses the SEERA (Software enginEERING in SudAn) cost estimation dataset, a collection of 120 software development projects from 42 organizations in Sudan. Unlike current cost estimation datasets, the SEERA dataset contains 76 attributes and is augmented with metadata and raw data. The SEERA data used in this study is thoroughly described in (Al Asheeri & Hammad, 2019). The paper discusses the data collection process, submitting organizations, and project characteristics.

The dataset was specifically collected to better represent software development projects facing cost and time constraints common in developing country contexts, like Sudan. In evaluating its suitability, we considered characteristics, like the sample size, number, and relevance of project features collected, representation of local industry factors, and documentation regarding data collection procedures. Compared to other publicly available datasets, SEERA captured the unique challenges and trade-offs faced by software teams operating under constrained budgets and deadlines.

The authors in (Al Asheeri & Hammad, 2019) also provide a general analysis of the dataset projects, illustrating the impact of local factors on software project cost and comparing the data quality of the SEERA dataset to public datasets from the PROMISE repository.

The SEERA dataset significantly contributes to the diversity of cost estimation datasets, filling the gap in constrained environment representation. Researchers can use the SEERA dataset to develop new cost estimation techniques that are more suitable for these environments and to evaluate the

generalization of previous methods. Ultimately, the SEERA dataset can improve the accuracy and effectiveness of software cost estimation in constrained environments.

For the regression models, we chose algorithms commonly used in previous software cost estimation studies, including Linear Regression, Decision Trees, Support Vector Machines, Neural Networks, and ensemble methods. This allowed for comparing SEERA-based results to prior work, while exploring a variety of model types with different strengths.

Table 4 provides information about the methodologies used in a sample of 120 software development projects. The most common methodology used was the Hybrid methodology, employed in 42 projects, representing 35% of the total. The Waterfall model was the second most used methodology, with 41 projects, making up 34% of the total. The Agile model was used in 27 projects, representing 23% of the total. The remaining methodologies, Prototyping, no methodology, and others, were used in 5%, 3%, and 1% of the projects, respectively. The Table’s final column, Total, indicates that 120 software development projects were included in the sample. The information in this Table can be used to analyze the characteristics of different software development methodologies and assist in making informed decisions regarding software development budgets and timelines based on the results of the cost estimation predictions.

The type of application domain influences the choice of software development model. For example, the Waterfall model is often used in safety-critical domains, such as aerospace and medical devices, because it provides a structured and predictable approach to development. The requirements are typically well-defined and unchanging. On the other hand, in domains such as e-commerce and web development, the Agile model is more commonly used because it is more flexible and allows for iterative development to quickly adapt to changing market demands. Similarly, in game development, the iterative and collaborative nature of the Agile model is preferred due to the constant need for feedback and adaptation to meet player expectations. The choice of development model also impacts the cost estimation process, with the Agile model typically requiring more frequent updates to the cost estimates due to the iterative and evolving nature of development. Therefore, the application domain and the associated software development model must be considered when developing cost estimation models to ensure they are appropriate for the specific development environment—Table 5 shows the project development type and application domain in the SEERA dataset.

### 3.2 CE-SWM-PRED: Approach

This section presents the CE-SWM-PRED, a machine learning-based approach for predicting cost estimation under different software development approaches in a constrained environment. The CE-SWM-PRED utilizes eight different machine learning algorithms, including Artificial Neural Networks (ANN), Support Vector Machines (SVM), Random Forests (R.F.), Generalized Linear

Table 4. Software development methodologies in SEERA dataset

Methodology	# of Projects	%
HYBRID METHODOLOGIES	42	35%
WATERFALL	41	34%
AGLE	27	23%
PROTOTYPING	5	4%
NO METHODOLOGY	4	3%
OTHER	1	1%
TOTAL	120	100%



Table 5. Project development type and application domain

Development	Application Domain						#	%
	Bespoke Application	ERP	Financial and Managerial	Banking Systems	Web Applications	Mobile Applications		
NEW SOFTWARE DEV.	32	16	12	14	10	6	90	75
CUSTOMIZATION OF IMPORTED SOFTWARE	2	9	-	-	-	-	11	
UPGRADING EXSISTING SOFTWARE	2	3	4	-	-	-	9	100
MODIFYING EXISTING SOFTWARE	2	6	2	-	-	-	10	32
#	38	34	18	14	10	6	120	28
%	32	28	15	12	8	5	100	

Models (LR), Elastic Net (EN), Ridge Regression (R.R.), Lasso Regression (LSR), and RANSAC Regressor (RAN). The steps involved in the CE-SWM-PRED are summarized as follows:

1. Read the dataset: The experimental data described in the previous section is read into the system.
2. Segment the SEERA dataset into three sub-datasets, each representing one of the software development approaches (Waterfall, Agile, Hybrid).
3. Preprocessing stage: The data is normalized to ensure that the values of the different variables are on the same scale.
4. Data split: The data is divided into a training set (70%) and a testing set (30%) to evaluate the performance of the models.
5. Model training: The eight machine learning algorithms are applied to the training data to build models.
6. Model testing: The trained models are tested using the testing data.
7. Evaluation metrics: Each model's performance is evaluated using metrics, such as mean absolute error, root mean squared error, and R-squared.
8. Repeat the process: Steps 3 to 6 are repeated 30 times to ensure the results are robust and reliable.

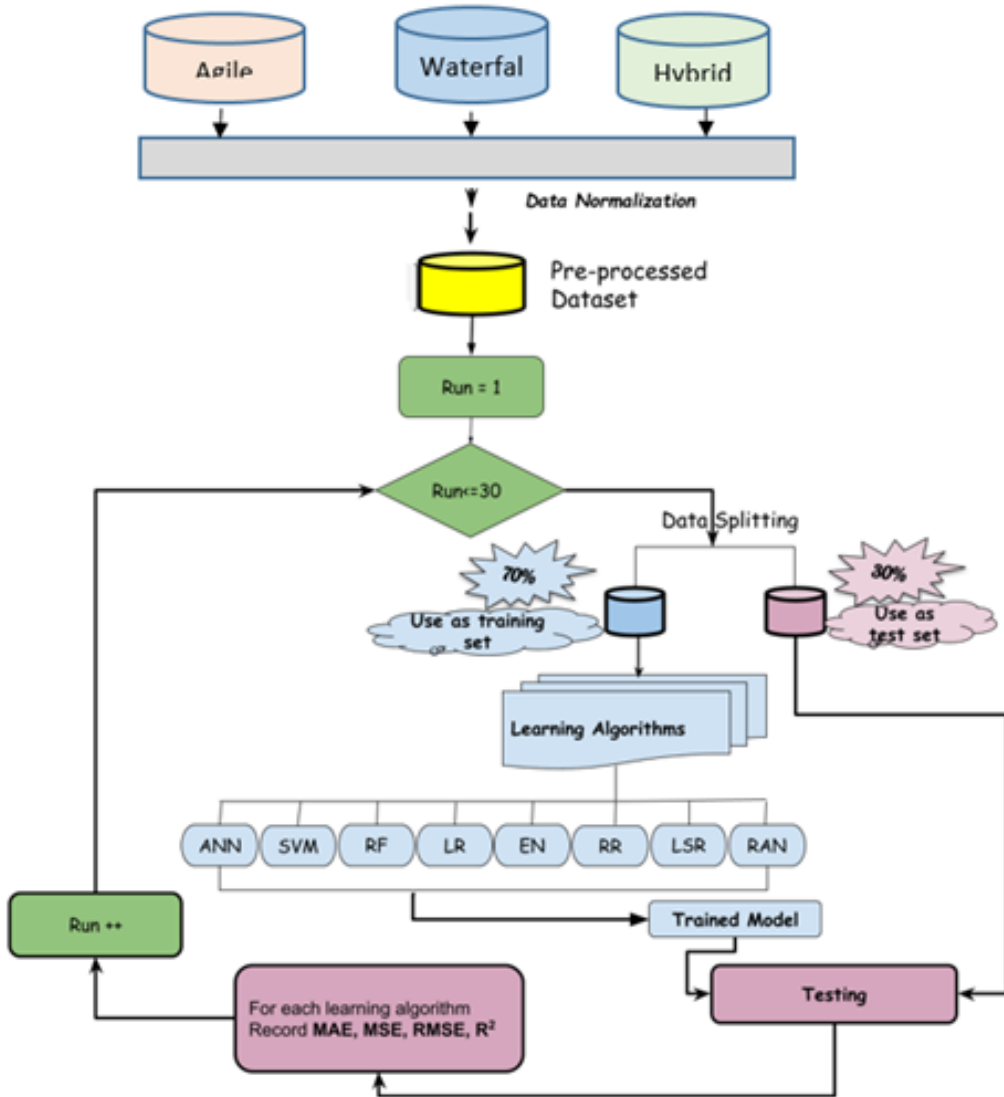
Fig . 2 depicts the procedure of the proposed approach, CE-SWM-PRED. The CE-SWM-PRED is designed to provide an accurate prediction of the cost estimation based on the input features.

### 3.2.1 Applied Regression Algorithms

**SVM:** Support Vector Machines (SVMs) are a machine learning algorithm for classification and regression. They were developed by Vapnik in 1995 and are based on statistical learning theory. The main idea behind SVM classification is to find the linear classifier that separates the classes with the most significant margin, meaning the enormous gap between the two classes, as defined by the distances from the closest points in each class to the hyperplane. In cases where a single hyperplane cannot separate the two classes, SVM will try to find the hyperplane that balances the margin and the number of misclassifications. It chooses a positive constant that trades off between the margin and the misclassification error, ensuring the optimal balance between the two. SVMs can also be used for nonlinear decision surfaces by mapping the original variables into a higher-dimensional feature space, and then, defining a linear classification problem. It allows SVMs to handle complex relationships between variables and targets, providing a flexible and effective solution for many real-world problems.

**Ridge Regression:** The Ridge Regression (R.R.) algorithm is a linear regression model that seeks to minimize the residual sum of squares (RSS) between the observed and predicted response values,

Figure 2. CE-SWM-PRED ALGO



while at the same time avoiding overfitting the data by adding a penalty term to the coefficients. The penalty term, alpha ( $\alpha$ ), is a hyperparameter that controls the strength of the regularization. Ridge Regression is a regularized linear regression method that addresses multicollinearity among predictors. In Ridge Regression, a penalty term is added to the least squares cost function to shrink the coefficients of the predictor variables toward zero. The objective function of Ridge Regression can be formulated as follows:

$$\min_{\beta} \|y - X\beta\|_2^2 + \alpha * \|\beta\|_2^2$$

Where  $\beta$  is the vector of coefficients,  $X$  is the design matrix,  $y$  is the target vector, and  $\alpha$  is the regularization parameter, which controls the magnitude of the penalty term. The R.R. algorithm is

a powerful tool for linear regression that allows us to account for multicollinearity and overfitting, while still obtaining meaningful predictions.

**Random Forest:** Random Forest (R.F.) is an ensemble machine learning algorithm widely used for regression and classification problems. It is an improvement over decision trees, which are prone to overfitting, by aggregating the results of many individual trees to make a prediction. R.F. works by constructing multiple decision trees using bootstrapped samples of the data and a random subset of the features at each split. The trees are constructed independently, and their predictions are combined by taking the mean or mode, depending on the problem being solved. This process makes a prediction less susceptible to overfitting than a single decision tree. The two main parameters in R.F. are the number of trees in the forest and the number of variables considered at each split in the tree. The number of trees in the forest can be increased to reduce variance, but this can increase the computational time and memory requirements. The number of variables considered at each split can be increased to reduce bias, which can also lead to overfitting the data. It is essential to tune these parameters through cross-validation and testing to find the optimal balance between variance and bias for a given data set.

**RANSACRegressor:** RANSAC, or the RANdom SAMple Consensus algorithm, is a robust regression method that fits a model to data containing outliers. The method works by randomly selecting a subset of the data and using it to fit the model. The model is then evaluated based on the number of points (the support size) with residual errors lower than a threshold value. The goal is to maximize the support size and minimize the outliers' effect on the model. The main parameters in the RANSACRegressor are *max\_trials*, which determines the maximum number of iterations RANSAC will perform before it terminates and is used to control the computational cost of the algorithm, and *residual\_threshold*, which determines the maximum residual error that a data point can have and still be considered an inlier. Points with residual errors more significant than the threshold are considered outliers and are not included in the model fit. The residual error is the difference between a given data point's predicted and actual values.

**Artificial Neural Network:** Artificial Neural Networks (ANNs) are a type of machine learning model inspired by the human brain's structure and function. In the context of regression, ANNs are used to model the relationship between a set of inputs and a continuous output variable. The basic building block of an ANN is the artificial neuron, which is a mathematical function that receives input from other neurons, processes it, and outputs a value. Multiple neurons are connected to form a network that can learn complex relationships between inputs and outputs. The training process of an ANN involves adjusting the weights of the connections between neurons so that the output of the network matches the target values, as closely as possible, using an optimization algorithm, such as gradient descent. The main parameters that can be adjusted in an ANN include the number of hidden layers, the number of neurons in each hidden layer, the activation function used by the neurons, the learning rate, and the optimization algorithm used to adjust the weights. The choice of these parameters can significantly impact the network's performance, so it is essential to experiment with different combinations to find the best values for a specific dataset.

**Lasso Regression:** Lasso Regression is a statistical technique used to perform linear regression. It works by utilizing the concept of shrinkage, where data values move closer to a central point, such as the mean. The Lasso Regression is particularly well-suited for models that contain a significant amount of multicollinearity, as it can help to eliminate noise and select only the most relevant variables. To achieve this, Lasso Regression adds limitations to the standard least squares (L.S.) technique. The Lasso method reduces the solution of L.S. to zero, or close to zero, for the coefficients of variables that are not as significant. As a result, it enables Lasso to act as a variable selection technique, resulting in a more straightforward and interpretable model. However, it is essential to note that the estimation obtained through Lasso Regression is subject to some bias. Lasso Regression has been used in previous studies to analyze data with many variables that explain the data. The regression coefficients produced by Lasso Regression are better equipped to pick explanatory factors than

those produced by traditional regression methods. Additionally, Lasso Regression can help address multicollinearity issues in regression analysis. When applied to data that contains grouped variables, the original form of Lasso Regression may not achieve the desired level of precision. In response to this limitation, researchers Yuan and Lin developed a novel strategy called Group Lasso, which improves the precision of Lasso Regression when applied to grouped variables.

**Elastic Net:** Elastic Net Regression, or ELNET, is a statistical technique that combines the regularization methods of Ridge Regression and Lasso Regression. This combination of methods allows ELNET to handle situations with high multicollinearity between the predictor variables, which leads to problems in traditional regression models, as the predictors can be highly correlated and impact the accuracy of the predictions. Hoerl and Kennard first introduced ELNET in 1970. Since then, it has been widely used to regularize data and select the most significant predictor variables to simplify the model and improve the accuracy of predictions. ELNET can eliminate or choose highly correlated predictor variables, leading to more accurate predictions. In addition, ELNET can overcome the limitations of time series variables used in regression analysis. When the behavior of time series variables is nonstationary and nonlinear, or when there is a problem with multicollinearity, ELNET can provide a solution. ELNET involves the decomposition of the original multivariate time-series predictors and the examination of their impact on the response variable, which helps to combat the correlation between the predictors. ELNET can eliminate or choose predictor variables that have a high level of correlation in the final model, which improves the accuracy of the predictions (Liu and Li, 2017).

The optimal selection of hyperparameters and variables is crucial in improving the prediction models' performance and ensuring the results' accuracy. Further experimentation and evaluation may lead to better results and an improved understanding of the relationships between the variables and the compressive strength of concrete. In the SVM model, the radial basis function (RBF) was used as the kernel function, costing 1000000 and an epsilon of 0.001. The Random Forest model used 1000 decision trees. In the Neural Network model, two hidden layers were used, each with 100 nodes and the maximum iteration was set to 1000. The Elastic Net model used an alpha value of 0.1, an l1 ratio of 0.9, and random selection.

### 3.3 Evaluation Metrics

We used various famous regression metrics to evaluate the performance of these algorithms (Brown, 2018). CE-SWM-PRED uses MAE, MSE, RMSE, and R2 to evaluate the quality of the different created models and compare their performance. MAE (Mean Absolute Error) measures the average magnitude of the errors in a set of predictions, without considering their direction. It is calculated as the average of the absolute differences between prediction and actual observation over the test sample. On the other hand, MSE (Mean Squared Error) is the average of the squared differences between the predicted and actual values, giving more weight to significant errors and making it more sensitive to outliers. RMSE (Root Mean Squared Error), which is the square root of the MSE, provides a measure of the absolute fit of the model to the data and is interpreted as the standard deviation of the unexplained variance. Finally, R2 (Coefficient of Determination) is a statistic that provides the proportion of the variation in the response variable, explained by the model's predictor variables. It ranges between 0 and 1, with a higher value indicating a better fit. These metrics provide valuable information about the machine learning algorithms' performance and help determine which algorithm best suits a given problem (Kaur, 2020).

**MSE (Mean Squared Error)**- The mean squared error is a standard measure of how well a model fits a dataset. It is the average of the squared differences between the predicted and actual values. MSE measures the average variance of the errors or the average squared deviation of the predictions from the actual values.

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

**MAE** (Mean Absolute Error)- The mean absolute error measures how well a model fits a dataset. It is the average of the absolute differences between the predicted and actual values. MAE measures the average magnitude of the errors or the average absolute deviation of the predictions from the actual values.

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

**RMSE** (Root Mean Squared Error)- The root mean squared error measures how well a model fits a dataset. It is the square root of the average of the squared differences between the predicted and actual values. RMSE measures the average variance of the errors or the average squared deviation of the predictions from the actual values.

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

**R-squared**- R-squared measures how well a model fits a dataset. It is the proportion of the variance in the dependent variable that the model explains. It measures how close the data are to the fitted regression line. It is also known as the coefficient of determination, or the coefficient of multiple determination for multiple regression.

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

#### 4. RESULTS AND DISCUSSION

Tables 6-8 present the evaluation results of various learning algorithms (ANN, SVM, R.F., L.R., EN, R.R., LSR, RAN) for the three Agile software development models based on different measures (MAE, MSE, RMSE, R2).

From Table 6, which represents the models based on the data set of the Agile software development approach, it can be observed that the Random Forest (R.F.) algorithm had the highest maximum MAE and MSE values. In contrast, the Artificial Neural Network (ANN) had the lowest minimum MAE and MSE values. Similarly, the R.F. algorithm had the highest maximum RMSE value, while the Support Vector Machine (SVM) had the lowest minimum RMSE value. The Average MAE, MSE, and RMSE values were the lowest for the ANN algorithm.

Regarding the coefficient of determination (R2), the ANN and SVM algorithms had the highest maximum R2 values. In contrast, the Ridge Regression (R.R.) and Randomized Lasso Regression (RAN) algorithms had the lowest minimum R2 values. The average R2 values were highest for the ANN and SVM algorithms. The standard deviation (Stdev) values indicate the variability of the evaluation results for each algorithm. The R.F. and EN algorithms had the highest Stdev values for MAE, MSE, and RMSE, indicating more significant variability in their performance. On the other

Table 6. Evaluation results of learning algorithms for agile sub-dataset (ANN, SVM, R.F., L.R., EN, R.R., LSR, RAN)

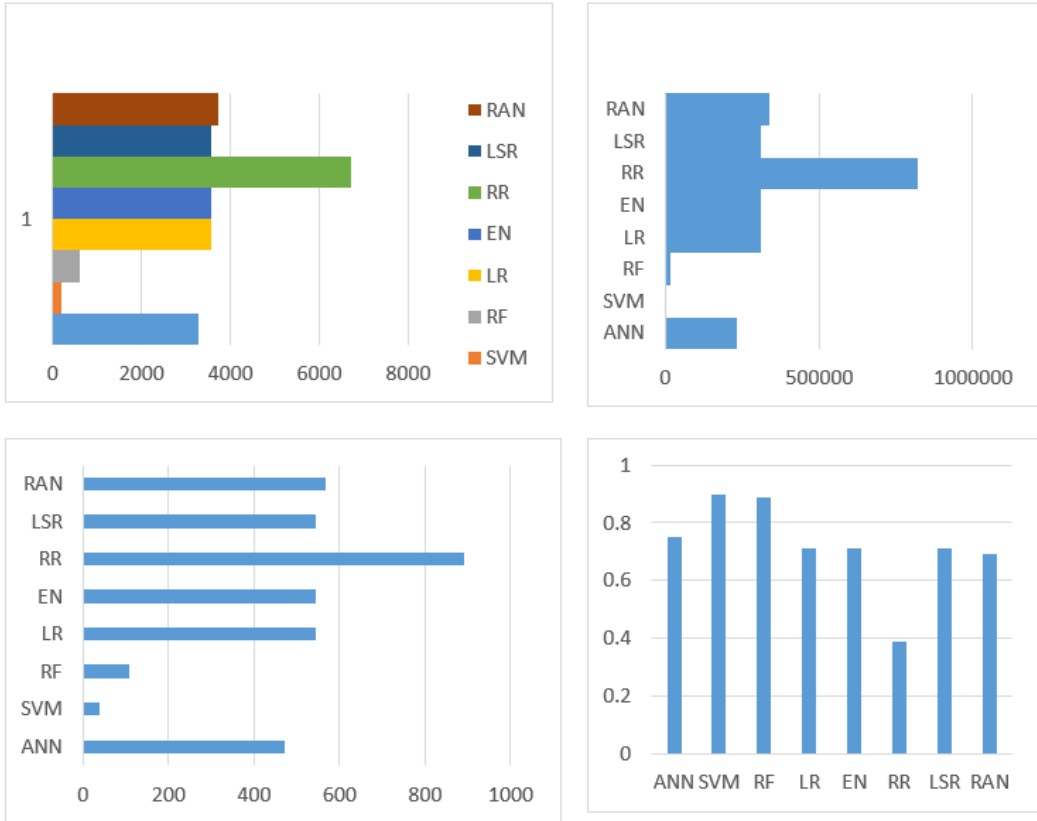
Measure	Learning Alog.							
	MAE							
	ANN	SVN	RF	LR	EN	RR	LSR	RAN
MAX	4343.5	618.31	1335.17	4687.33	4682.95	7953.35	4688.79	4745
MIN	2360.09	89.06	332.88	2618.51	2423.6	5177.16	2444.04	2817.8
AVERAGE	3291.57	191.99	610.28	3581.38	3577	6718.19	3585.03	3716.43
STDEV	538.01	149.65	246.01	498.59	518.3	685.47	512.46	506.62
MSE								
MAX	372980.36	16484.13	91677.05	450069.82	450539.94	1097557.19	450125.3	531425.4
MIN	120460.22	121.91	3031.69	152837.91	137399.14	510406.51	139223.41	135859.57
AVERAGE	232548.8	3215.65	15284.74	309122.15	309015.57	822731.17	309249.17	338104.61
STDEV	66092.74	4919.47	18786.55	75962.34				
RMSE								
MAX	603.298348	126.854703	299.117144	662.716652	663.054256	1034.92506	662.716652	720.109332
MIN	342.836862	10.887729	54.354244	386.218976	366.131538	705.761162	368.579167	364.105914
AVERAGE	471.632788	38.993262	108.792889	545.061658	544.724054	892.11857	544.977257	568.356334
STDEV	68.027206	40.850084	56.379868	68.871216	70.89684	84.823005	70.306033	84.569802
R <sup>2</sup>								
MAX	0.83	0.9	0.9	0.8	0.81	0.49	0.81	0.81
MIN	0.69	0.89	0.85	0.61	0.61	0.29	0.61	0.49
AVERAGE	0.75	0.9	0.89	0.71	0.71	0.39	0.71	0.69
STDEV	0.06	0.1	0.09	0.05	0.05	0.06	0.05	0.02

hand, the SVM algorithm had the lowest Stdev values for MAE, MSE, and RMSE, indicating a more consistent performance.

Based on the evaluation results presented in Table 6 and shown in Fig . 3, the ANN and SVM algorithms are the most promising learning algorithms for Agile software development regarding their overall performance across different measures.

Table 7 presents the results of different learning algorithms used for a regression problem, evaluated using a waterfall dataset based on different performance metrics. For more readability, the Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R2 score are also shown in Figure 4. Upon analyzing the results, we can make the following observations: The MAE metric measures the average absolute difference between the predicted and actual values. The lowest MAE value is achieved by the SVM learning algorithm (75.64), followed by R.F. (282.72), LSR (2075.76), ANN (3689), L.R. (3977.3), RAN (4030), R.R. (4397.04), and EN (6754.9). Therefore, SVM is the best algorithm for minimizing MAE, while EN is the worst. The MSE metric measures the average squared difference between the predicted and actual values. The lowest MSE value is achieved by the SVM learning algorithm (103.54), followed by R.F. (2574.86), LSR (118244.54), ANN (316777.84), L.R. (382251.08), RAN (382298.2), R.R. (433495.94), and EN (932171.86). Therefore, SVM is the best algorithm for minimizing MSE, while EN is the worst. The RMSE metric measures the square root of the average squared difference between the predicted and actual values. The lowest RMSE value is achieved by the SVM learning algorithm (10.15747032), followed by R.F. (50.70861152), LSR (343.8579294), ANN (562.8340918), L.R. (618.2671082), RAN (618.2671082), R.R. (658.424549), and EN (965.510861).

Figure 3. Summary of fitting indicators for the eight models used: Mean absolute error (MAE), mean square error (MSE), and the square root of the mean of the square of all of the errors (RMSE)



Therefore, SVM is the best algorithm for minimizing RMSE, while EN is the worst. The R2 score metric measures the proportion of the variance in the target variable that is predictable from the independent variables. The highest R2 score is achieved by the SVM learning algorithm (0.91), followed by R.F. (0.91), LSR (0.82), ANN (0.84), L.R. (0.81), RAN (0.82), R.R. (0.62), and EN (0.5). Therefore, SVM and R.F. are the best algorithms for maximizing the R2 score, while EN is the worst. The SVM learning algorithm appears to be the best-performing overall, as it achieves the lowest values for MAE, MSE, and RMSE and the highest R2 score. Table 8 presents the evaluation results of learning algorithms using the Hybrid software development approach dataset. The SVM algorithm has the lowest MAE, indicating that it performs better at predicting the target variable.

In contrast, the ANN algorithm has the highest MAE, implying the most considerable absolute difference between the predicted and actual values. The R.F. algorithm has the highest MSE, indicating that it performs the worst in predicting the target variable. The EN algorithm has the lowest MSE, implying a minor squared difference between the predicted and actual values. The SVM algorithm has the lowest RMSE, indicating that it performs better predicting the target variable. The ANN algorithm has the highest RMSE, implying the most significant difference between the predicted and actual values. The EN algorithm has the highest R2, indicating that it explains the most variance in the target variable. The R.R. and LSR algorithms have the lowest R2, implying that they explain the minor variance in the target variable. Moreover, the results in Figure 5 suggest that the SVM and EN algorithms best predict the target variable, while the R.F. algorithm performs the worst. However,

Table 7. Evaluation results of learning algorithms waterfall sub-dataset (ANN, SVM, R.F., L.R., EN, R.R., LSR, RAN)

Measure	Learning Alog.							
	MAE							
	ANN	SVN	RF	LR	EN	RR	LSR	RAN
MAX	3689	525.14	1133.98	3981.02	3977.3	6754.9	3982.26	4030
MIN	2004.46	75.64	282.72	2223.94	2058.4	4397.04	2075.76	2393.2
AVERAGE	2795.58	163.06	518.32	3041.72	3038	5705.86	3044.82	3156.42
STDEV	456.94	127.1	208.94	423.46	440.2	582.18	435.24	430.28
MSE								
MAX	316777.84	14000.22	77862.7	382251.08	382650.36	932171.86	382298.2	451347.6
MIN	102308.68	103.54	2574.86	129807.54	116695.16	433495.94	118244.54	115387.58
AVERAGE	197507.2	2731.1	12981.56	262542.1	262451.58	698757.98	262649.98	287157.34
STDEV	56133.56	4178.18	15955.7	64515.96	65798.12	129310.92	65297.78	83184.78
RMSE								
MAX	562.83	118.35	279.05	618.27	618.58	965.51	618.27	671.81
MIN	342.836862	10.887729	54.354244	386.218976	366.131538	705.761162	368.579167	364.105914
AVERAGE	471.632788	38.993262	108.792889	545.061658	544.724054	892.11857	544.977257	568.356334
STDEV	68.027206	40.850084	56.379868	68.871216	70.89684	84.823005	70.306033	84.569802
R <sup>2</sup>								
MAX	0.83	0.9	0.9	0.8	0.81	0.49	0.81	0.81
MIN	0.69	0.89	0.85	0.61	0.61	0.29	0.61	0.49
AVERAGE	0.75	0.9	0.89	0.71	0.71	0.39	0.71	0.69
STDEV	0.06	0.1	0.09	0.05	0.05	0.06	0.05	0.02

the choice of the algorithm may also depend on other factors, such as the computational complexity and interpretability of the model.

Based on the evaluation results presented in Tables 6-8 and Figure 3, 4, and 5, it can be concluded that the performance of the learning algorithms varies depending on the dataset used and the performance metric evaluated. However, the SVM and ANN algorithms appear to be the most promising algorithms for Agile software development. In contrast, the SVM algorithm best predicts the target variable across all three software development approaches. It is important to note that the choice of an algorithm may also depend on other factors such as interpretability, computational complexity, and the specific problem being addressed.

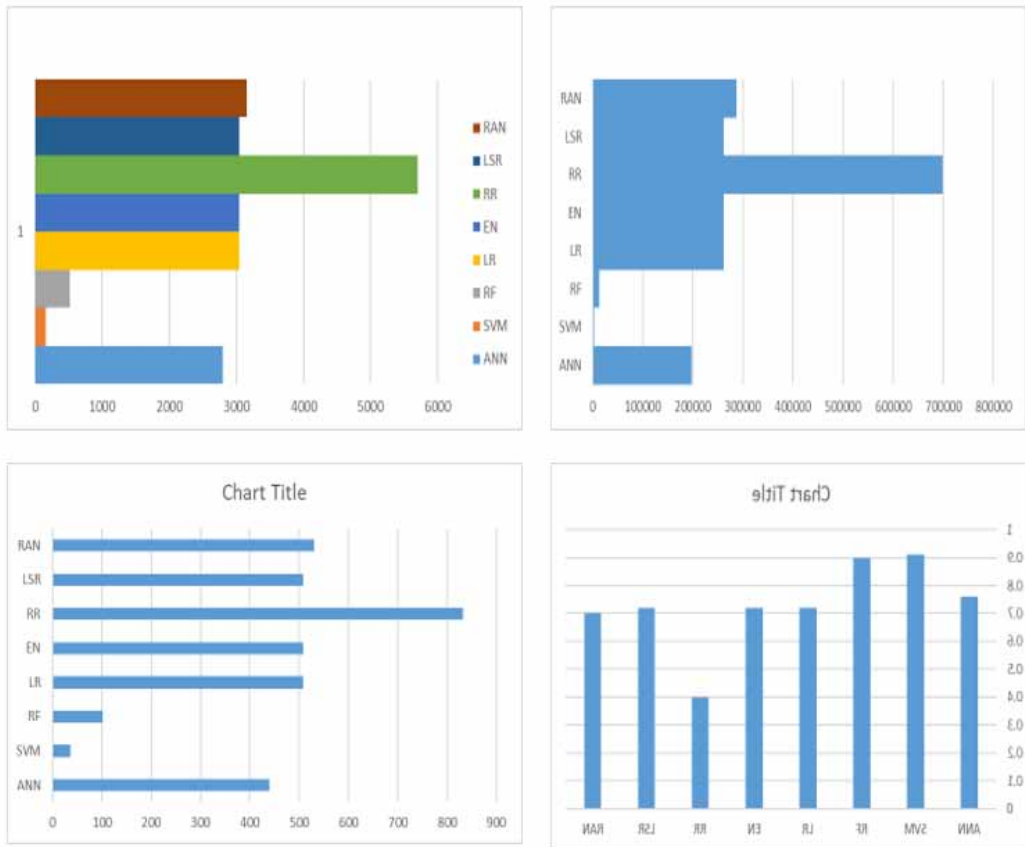
Looking at the values in Table 9 for the AME of cost estimation for the three software development approaches, we can see that SVM has the lowest MAE values across all three software development models. It suggests that SVM has the best performance in terms of accuracy for cost estimation.

In terms of the different software development models, the Waterfall model has the lowest average MAE value (2795.58), compared to the Hybrid (3066.12) and Agile (3291.57) models. The results suggest that the cost estimation performance for the Waterfall model is better, compared to the other two models. It is important to note that the MAE values for all the models are relatively high, indicating room for improvement in the cost estimation process, regardless of the software development model used.

Looking at the RMSE values for the three software development models, we can see that SVM has the lowest RMSE value in all three cases, which indicates that SVM is the most accurate model



Figure 4. Summary of fitting indicators for the eight models used: Mean absolute error (MAE), mean square error (MSE), and the square root of the mean of the square of all of the errors (RMSE)



for cost estimation among the four models. The other models have similar RMSE values, with R.F. and L.R. having slightly lower values than the others.

It is worth noting that the RMSE values are relatively high in absolute terms, indicating that the model's predictions may still have a significant margin of error. Nonetheless, the relative differences between the models suggest that SVM is the most accurate of the models presented here.

The overall result for all evaluation metrics is shown in Figure 6.; when selecting a model for software development cost estimation, it is essential to consider the R-squared and RMSE values and other factors, such as model interpretability, data availability, and computational efficiency. Moreover, we can see that the SVM model has the lowest MSE for all three software development models, followed closely by the Random Forest (R.F.) model. These models can estimate the cost of software development projects more accurately than the other models. On the other hand, the Ridge Regression (R.R.) and Elastic Net (EN) models have the highest MSE values for all three software development models. They are less accurate in estimating the cost of software development projects. The R2 value is a statistical measure that indicates the proportion of the variation in the dependent variable (i.e., cost) explained by the independent variables (i.e., features). In this case, we can see that the R2 values for the three software development models (Hybrid, Waterfall, and Agile) are relatively similar across all eight machine learning algorithms. The highest R2 value is observed for the Support Vector Machine (SVM) and Random Forest (R.F.) algorithms for both the Waterfall and Agile models.

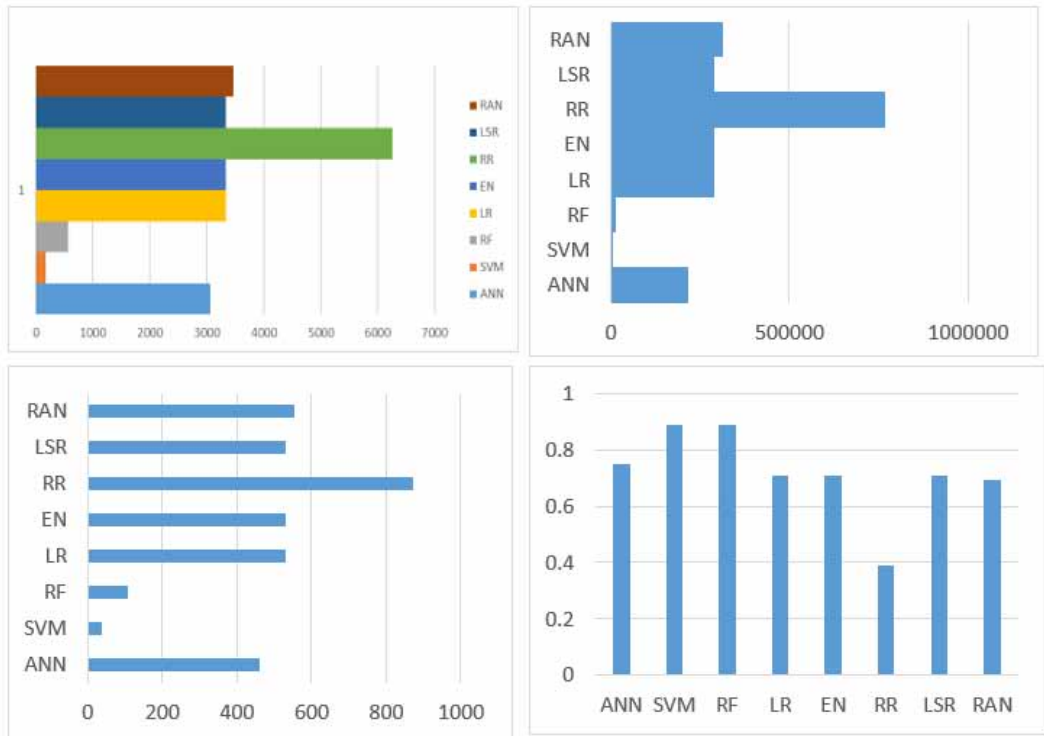
Table 8. Evaluation results of learning algorithms hybrid sub-dataset (ANN, SVM, R.F., L.R., EN, R.R., LSR, RAN)

Measure	Learning Alog.							
	MAE							
	ANN	SVN	RF	LR	EN	RR	LSR	RAN
MAX	4046	575.96	1243.72	4366.28	4362.2	7408.6	4367.64	44.20
MIN	2198.44	82.96	310.08	2439.16	2257.6	4822.56	2276.64	2624.8
AVERAGE	3066.12	178.84	568.48	3336.08	3332	6258.04	3339.48	3461.88
STDEV	501.16	139.4	229.16	464.44	482.8	638.52	477.36	471.92
MSE								
MAX	589.42	123.94	292.24	647.48	647.81	1011.12	647.48	703.55
MIN	113.56	2824.04	142369.56	127988.24	475447.16	129687.56	126554.12	2995.4
AVERAGE	2198.44	82.96	310.08	2439.16	2257.6	4822.56	2276.64	2624.8
STDEV	61565.84	4582.52	17499.8	70759.44	72165.68	72165.68	141824.88	91234.92
RMSE								
MAX	589.42	123.94	292.24	647.48	647.81	1011.12	647.48	703.55
MIN	334.95	10.64	53.10424	377.34	357.71	689.53	360.10	355.73
AVERAGE	460.79	38.10	106.290	532.52	532.1968	871.6022	532.44422	5555.28564
STDEV	66.46276	39.91064	55.08328	67.28736	69.2664	82.8723	68.68918	82.62
R <sup>2</sup>								
MAX	0.83	0.9	0.89	0.8	0.81	0.49	0.81	0.81
MIN	0.69	0.89	0.85	0.61	0.61	0.29	0.61	0.49
AVERAGE	0.75	0.89	0.89	0.71	0.71	0.39	0.71	0.69
STDEV	0.06	0.1	0.09	0.05	0.05	0.06	0.05	0.02

In contrast, the highest R2 value for the Hybrid model is observed for the SVM algorithm. Overall, the R2 values indicate that the machine learning algorithms can explain a significant portion of the variation in software model cost estimates, with R2 values ranging from 0.39 to 0.91. However, it is worth noting that the R2 values are not extremely high, indicating that there may still be some unexplained variation in the cost estimates. Additionally, the R2 values alone do not provide information about the cost estimates' accuracy. Other performance metrics, such as Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE), should also be considered when evaluating the performance of the machine learning algorithms.

Selecting the appropriate software development model for cost estimation in a constrained environment requires careful consideration of project scope, requirements, and available resources. Each model has its benefits and drawbacks, and choosing the suitable model can significantly impact the accuracy and reliability of cost estimation. A Hybrid model may be a good option, as it provides a balance of flexibility and structure while considering the constraints of the project. Cost estimation is an essential part of software development, and it is crucial to choose an appropriate software development model for accurate and reliable cost estimation. Choosing a suitable software development model can be challenging in a constrained environment with limited resources. Among the different software development models, Waterfall, Agile, and Hybrid are popular options for cost estimation. The Waterfall model is a linear sequential approach that follows a strict order of development phases, which can be helpful in small projects with a well-defined scope and requirements. However, in

Figure 5. Summary of fitting indicators for the eight models used: Mean absolute error (MAE), mean square error (MSE), and the square root of the mean of the square of all of the errors (RMSE)



constrained environments, it may not be practical, as it requires a lot of planning and documentation, which can be time-consuming and costly.

Conversely, the agile model is an iterative approach focusing on flexibility, adaptability, and collaboration between the development team and the client. It can be helpful in constrained environments, as it allows for changes to be made to the project scope and requirements based on client feedback, which can help save time and resources. However, estimating costs accurately in an Agile model can also be challenging, as the scope and requirements change frequently. The Hybrid model combines Waterfall and Agile elements, which can be beneficial in constrained environments. It allows for flexibility and adaptability, while maintaining some of the structure and planning of the Waterfall model. It can also be helpful in projects with evolving requirements and an uncertain scope. However, it can be challenging to estimate costs accurately in a Hybrid model, as it requires balancing the benefits of both Waterfall and Agile models, while accounting for their drawbacks. Cost estimation becomes critical to project planning and management in constrained environments. Cost estimation involves predicting the resources needed for a project, including time, money, and personnel. Accurate cost estimation is critical to ensure a project can be completed within the allocated budget and timeline. In practical terms, several factors should be considered when estimating the cost in constrained environments. One of the most important is the accuracy of the estimates. Estimation techniques, such as parametric estimation, analogous estimation, and expert judgment can be used to develop accurate cost estimates. It is also essential to consider the level of uncertainty in the estimates and to use contingency planning to account for unexpected events that may impact project costs. Other factors to consider are cost, scope, and schedule trade-offs. In constrained environments, it is often necessary to balance the resources available with the project's scope and the completion timeline,

Table 9. Software development mode

Software Dev. Mode	Learning Algo.							
	ANN	SVM	RF	LR	EN	RR	LSR	RAN
	MAE							
HYBRID	3066.12	178.84	568.48	3336.08	3332	6258.04	3339.48	3461.88
WATERFALL	2795.58	163.06	518.32	3041.72	3038	5705.86	3044.82	3156.42
AGILE	3291.57	191.99	610.28	3581.38	3577	6718.19	3585.03	3716.43
MSE								
HYBRID	216620.8	2995.4	14237.84	287949.4	287850.1	766379.7	288067.7	314946.8
WATERFALL	197507.2	2731.1	12981.56	262542.1	262451.6	698758	262650	287157.3
AGILE	232548.8	3215.65	15284.74	309122.2	309015.6	822731.2	309249.2	338104.6
RMSE								
HYBRID	460.7865	38.09652	106.2909	532.5267	532.1968	871.6022	532.4442	555.2856
WATERFALL	439.9996	36.37792	101.496	508.5034	508.1885	832.2826	508.4247	530.2357
AGILE	471.63	38.99326	108.7929	545.0617	544.7241	892.1186	544.9773	568.356
R <sup>2</sup>								
HYBRID	0.75	0.89	0.89	0.71	0.71	0.39	0.71	0.69
WATERFALL	0.76	0.91	0.9	0.72	0.72	0.4	0.72	0.7
AGILE	0.75	0.9	0.89	0.71	0.71	0.39	0.71	0.69

which requires making difficult decisions about project priorities, such as reducing the project scope or extending the timeline for completion. Finally, effective communication and collaboration among team members are essential for cost estimation in constrained environments. Project stakeholders, including team members, sponsors, and clients, must be involved in the estimation process and kept informed of any changes or updates to the estimates to ensure that everyone is on the same page and that there are no surprises later in the project lifecycle. In summary, cost estimation in constrained environments requires a careful balance between accuracy, scope, and timeline. Using effective estimation techniques, contingency planning, and communication and collaboration among team members, project managers can develop and manage cost estimates that help ensure their projects' success.

Cost estimation using different software development models requires practical considerations in constrained environments. Constrained environments refer to limited resources, such as time, budget, or personnel. One practical consideration is carefully evaluating the chosen model's ability to adapt to project scope or requirement changes. Agile and Hybrid development models are more suited to adapt to changes because they are iterative and focus on continuous improvement, allowing for more flexibility in cost estimation as the project progresses. Another consideration is to ensure that the estimated costs align with the project's objectives and desired outcomes. In constrained environments, allocating resources efficiently and ensuring that the costs are not exceeding the available budget is crucial. Waterfall development models are typically more rigid and may not allow cost adjustments during the project's lifecycle. Hence, it is essential to evaluate the project's requirements carefully before choosing the model for cost estimation. Finally, in constrained environments, monitoring the project's progress and adjusting the cost estimation is vital for implementing effective project management practices and tools that allow real-time tracking of project costs, risks, and timelines. Regularly updating and reviewing cost estimation using different software development models can

Figure 6. Summary of fitting indicators for the eight models used: Mean absolute error (MAE), mean square error (MSE), and the square root of the mean of the square of all of the errors (RMSE)



help ensure the project remains on track and within budget. Evaluating the chosen model's ability to adapt to changes, ensuring that the estimated costs align with the project's objectives, and regularly monitoring the project's progress are all critical factors to consider. With proper planning and project management, organizations can develop accurate cost estimates that align with their constraints and deliver successful projects.

The benefits of cost estimation using machine learning models for industries in developing countries with constrained environments are significant. Firstly, machine learning models can handle large volumes of data, enabling accurate cost estimation, which is instrumental to developing countries with limited resources. Data collection can be challenging due to inadequate infrastructure. Machine learning models can learn from available data and use this knowledge to make predictions, making it easier to estimate costs and make informed decisions. Secondly, machine learning models can help industries in developing countries optimize their resources and minimize costs significant in constrained environments, where resources are scarce and every resource is critical. Machine learning models can analyze data and identify cost-saving opportunities, helping industries make better decisions on where to allocate their resources and optimize their operations, reduce waste, and improve overall efficiency. Lastly, cost estimation using machine learning models can help industries in developing countries increase their competitiveness by accurately estimating costs and optimizing their resources. Industries can offer their products and services at competitive prices, making them more attractive to customers, helping industries grow, creating jobs, and contributing to the country's economic development. In summary, cost estimation using machine learning models

can benefit industries in developing countries with constrained environments. By leveraging machine learning models, industries can accurately estimate costs, optimize their resources, and increase competitiveness, ultimately contributing to the country's economic development.

## 5. CONCLUSION

In conclusion, this paper aimed to evaluate the performance of various regression models for software development cost estimation in constrained environments. The study utilized the SEERA dataset, representing economically and technically constrained software industries. The eight regression models evaluated were Elastic Net, Lasso Regression, Linear Regression, Neural Network, RANSACRegressor, Random Forest, Ridge Regression, and SVM. The performance of these models was assessed using correlation coefficients and accuracy indicators. The results showed that SVM and Random Forest models best estimated software development costs in constrained environments. However, the Elastic Net, Lasso Regression, Linear Regression, Neural Network, and RANSACRegressor models also performed well. These findings provide insights into selecting appropriate regression models based on the specific software model and dataset used in the estimation process. The study highlights the importance of using relevant and quality datasets for accurate cost estimation in software development. The availability of datasets representing constrained environments is limited, and this paper fills a gap in the literature by using the SEERA dataset. The study also emphasizes software development industries' challenges in constrained environments, such as limited resources, infrastructure, and expertise.

In conclusion, the study provides valuable insights into software development cost estimation in constrained environments. It can help project managers and developers select appropriate regression models for cost estimation in these settings. Further research can expand on this work by incorporating additional datasets and exploring the use of other machine-learning techniques for cost estimation in constrained environments.

## ACKNOWLEDGMENT

The authors extend their appreciation to the Deanship of Scientific Research at Shaqra University for funding this research work through the project number (ANN-SU-2023036).

## REFERENCES

- Akbar, M. A., Sang, J., Nasrullah, , Khan, A. A., Shafiq, M., & Fazal-E-Amin, . (2019). Towards the guidelines for requirements change management in global software development: Client-vendor perspective. *IEEE Access : Practical Innovations, Open Solutions*, 7, 76985–77007. doi:10.1109/ACCESS.2019.2918552
- Al Asheeri, M. M., & Hammad, M. (2019). *Machine learning models for software cost estimation, International Conference on Innovation and Intelligence for Informatics*. Show in Context.
- Ali, A., & Gravino, C. (2021). Improving software effort estimation using bio-inspired algorithms to select relevant features: An empirical study. *Science of Computer Programming*, 205, 102621. doi:10.1016/j.scico.2021.102621
- Ali & Gravino. (2019). A systematic literature review of software effort prediction using machine learning methods. *Journal of Software (Malden, MA)*, 31(10).
- Boehm, B. W. (2017). *Software cost estimation meets software diversity*. In Proceedings of the 39th International Conference on Software Engineering Companion (ICSE-C'17), Buenos Aires, Argentina. doi:10.1109/ICSE-C.2017.159
- Casado-Lumbreras, C., Colomo-Palacios, R., Francisca, N., & Misra, S. (2014). Software development outsourcing: Challenges and opportunities in Nigeria. *Journal of Global Information Technology Management*, 17(4), 267–282. doi:10.1080/1097198X.2014.978626
- Chhabra, S., & Singh, H. (2020). Optimizing design of fuzzy model for software cost estimation using particle swarm optimization algorithm. *International Journal of Computational Intelligence and Applications*, 19(1), 2050005. doi:10.1142/S1469026820500054
- Di Martino, S., Ferrucci, F., Gravino, C., & Sarro, F. (2020). Assessing the effectiveness of approximate functional sizing approaches for effort estimation. *Information and Software Technology*, 123, 106308. doi:10.1016/j.infsof.2020.106308
- Emtinan, I., Mustafa, & Osman, R. (2020). SEERA: A software cost estimation dataset for constrained environments. In *Proceedings of the 16th ACM International Conference on Predictive Models and Data Analytics in Software Engineering (PROMISE 2020)*. Association for Computing Machinery.
- Holtkamp, P., Lau, I., & Pawlowski, J. M. (2015). How software development competencies change in global setting: An explorative study. *Journal of Software (Malden, MA)*, 27(1), 50–72. doi:10.1002/smr.1701
- Ilyas, M., Khan, S. U., & Rashid, N. (2020). Empirical validation of software integration practices in global software development. *SN Computer Science*, 1(3), 1–23. doi:10.1007/s42979-020-00175-2
- Jha, M., & Jha, R. (2020). Comparing the effort estimated by different models. 6th International Conference on Advanced Computing and Communication Systems (ICACCS), (pp. 1148-1154). ACM.
- Kanakasabhpathi, P., & Muthayyan, J. (2019). A real-time extreme learning machine for software development effort estimation. *The International Arab Journal of Information Technology*, 16(1).
- Kumar, P. S., Behera, H. S., Kumari, A., Nayak, J., & Naik, B. (2020). Advancement from neural networks to deep learning in software effort estimation: Perspective of two decades. *Computer Science Review*, 2020(38), 100–288.
- Kumar, P. S., Behera, H. S., Nayak, J., & Naik, B. (2021). A pragmatic ensemble learning approach for effective software effort estimation. *Innovations in Systems and Software Engineering*, 1, 1–17.
- Mahmood, Y., Kama, N., Azmi, A., Khan, A. S., & Ali, M. (2021). Software effort estimation accuracy prediction of machine learning techniques: A systematic performance evaluation. *J. Softw. Pract. Exp.*

Mohammed, M., & Jamal, S. A. (2022). An overview of machine learning approaches to software development cost estimation. *8th International Conference on Contemporary Information Technology and Mathematics (ICCITM)*, (pp. 153-158). IEEE.

Mukherjee, S., & Malu, R. K. (2014). *Optimization of project effort estimate using neural network*. In *Proceedings of the 2014 IEEE International Conference on Advanced Communications, Control and Computing Technologies*, Ramanathapuram, India. doi:10.1109/ICACCCT.2014.7019474

Nassif, A. B., Azzeh, M., Idri, A., & Abran, A. (2019). Software development effort estimation using regression fuzzy models. *Computational Intelligence and Neuroscience*, 2019, 1–17. doi:10.1155/2019/8367214 PMID:30915110

Nhung, H. L. T. K., Hoc, H. T., & Hai, V. V. (2019). A review of use case-based development effort estimation methods in the system development context. In *Intelligent Systems Applications in Software Engineering* (pp. 484–499). Springer. doi:10.1007/978-3-030-30329-7\_44

Panda, N., & Majhi, S. K. (2020). How effective is the salp swarm algorithm in data classification. In *Computational Intelligence in Pattern Recognition* (pp. 579–588). Springer. doi:10.1007/978-981-13-9042-5\_49

Pandey, M., Litoriya, R., & Pandey, P. (2020). Applicability of machine learning methods on mobile app effort estimation: Validation and performance evaluation. *International Journal of Software Engineering and Knowledge Engineering*, 2020(30), 23–41. doi:10.1142/S0218194020500023

Popović, J., & Bojić, D. (2012). A comparative evaluation of effort estimation methods in the software life cycle. *Computer Science and Information Systems*, 9(1), 455–484. doi:10.2298/CSIS110316068P

Promise Software Engineering Repository. (n.d.). *Home*. Promise Software.

Rankovic, D., Rankovic, N., Ivanovic, M., & Lazic, L. (2021). Convergence rate of artificial neural networks for estimation in software development projects. *Information and Software Technology*, 2021(138), 106627. doi:10.1016/j.infsof.2021.106627

Rankovic, N., Rankovic, D., Ivanovic, M., & Lazic, L. (2021). A new approach to software effort estimation using different artificial neural network architectures and Taguchi Orthogonal Arrays. *IEEE Access : Practical Innovations, Open Solutions*, 9, 26926–26936. doi:10.1109/ACCESS.2021.3057807

Saavedra Martínez, J. I., Valdés Souto, F., & Rodríguez Monje, M. (2020). Analysis of automated estimation models using machine learning. In *Proceedings of the 8th International Conference in Software Engineering Research and Innovation (CONISOFT)*. IEEE. doi:10.1109/CONISOFT50191.2020.00025

Safari, S., & Erfani, A. R. (2020). A new method for fuzzification of nested dummy variables by fuzzy clustering membership functions and its application in financial economy. *Iran. J. Fuzzy Syst.*, 17, 13–27.

Sakhrawi, Z., Sellami, A., & Bouassida, N. (2020). Investigating the impact of functional size measurement on predicting software enhancement effort using correlation-based feature selection algorithm and SVR method. In S. Ben Sassi, S. Ducasse, & H. Mili (Eds.), *Reuse in emerging software engineering practices (19th International Conference on Software and Systems Reuse)*. Springer. doi:10.1007/978-3-030-64694-3\_14

Salmanoglu, M., Hacaloglu, T., & Demirors, O. (2017). Effort estimation for agile software development: Comparative case studies using COSMIC functional size measurement and story points. *Proceedings of the 27th International Workshop on Software Measurement and 12th International Conference on Software Process and Product Measurement (IWSM Mensura' 17)*. Association for Computing Machinery.

Zhao, H., & Zhang, C. (2020). An online-learning-based evolutionary many-objective algorithm. *Information Sciences*, 509, 1–21. doi:10.1016/j.ins.2019.08.069