Tomas Bata University in Zlín
Faculty of Applied Informatics

Doctoral Thesis

Effective Parametric Model for System Engineering Project Estimation

Efektivní parametrický model pro odhad projektu systémového inženýrství

Author: **Ho Le Thi Kim Nhung**

Degree programme: Engineering Informatics

Degree course: Software Engineering

Supervisor: Assoc. Prof. Ing. Zdenka Prokopová, CSc.

Consulting supervisor: Assoc. Prof. Ing. Radek Šilhavý, Ph. D.

Zlín, October 2022

ACKNOWLEDGEMENT

I am grateful to my supervisor, Assoc. Prof. Ing. Zdenka Prokopová, CSc., as well as my consulting supervisors, Assoc. Prof. Ing. Radek Šilhavý, Ph. D. and Assoc. Prof. Ing. Petr Šilhavý, Ph. D., for their motivation, direction, and help throughout my Ph.D. program. Their constructive comments pushed me to do my research more efficiently and to publish our works in respected conferences and journals.

I would also like to thank my colleagues, friends, and family for their moral support during my studies.

ABSTRAKT

V předkládané disertační práci jsou představeny návrhy nových způsobů odhadů složitosti projektů založených na metodě Use Case Points, která se používá v raných fázích vývoje softwaru. Navržené metody jsou vyvinuty tak, aby zvládaly nepřesnosti při odhadování a zahrnovaly expertní posudky pro vytvoření přesných a spolehlivých odhadů úsilí. Každý přístup má své výhody a vzájemně se doplňují. Cílem je, aby jednotlivé metody vytvořily kompletní proces a podporovaly efektivitu odhadu úsilí, tj. aby se ve všech situacích účinněji minimalizovala chyba v odhadu. Výsledky ukazují, že navržené metody Software Development Effort Estimation (SDEE) jsou konkurenceschopné ve srovnání s jinými alternativami, na základě sedmi hodnotících kritérií a statistických párových srovnání t-testů.

Key words in Czech: *odhad úsilí vývoje softwaru, body případů užití, optimalizace korekčních faktorů*

ABSTRACT

In the presented doctoral thesis, proposals for new methods of estimating the complexity of projects based on the Use Case Points method, which is used in the early stages of software development, are presented. The proposed methods are developed to handle estimation inaccuracies and incorporate expert judgments to produce accurate and reliable effort estimates. Each approach has its advantages, and they complement each other. The goal is for them to create a complete process and support the efficiency of effort estimation, i.e., to minimize estimation error more effectively in all situations. The results show that the proposed Software Development Effort Estimation (SDEE) methods are competitive compared to other alternatives, based on seven evaluation criteria and statistical pairwise t-test comparisons.

Key words: *software development effort estimation, use case points, optimising correction factors*

CONTENTS OF THE THESIS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

Abbreviations	Description
SDEE	Software development effort estimation
UCP	Use Case Points
UCM	Use Case Model
TCF	Technical complexity factors
ECF	Environmental complexity factors
MLR	Multiple linear regression
ML	Machine learning
PF	Productivity factor
LASSO	Least Absolute Shrinkage and Selection Operator
OCF	Optimization Correction Factors
LSR	Least square regression
ExOCF	Extension of Optimizing Correction Factors
KNN	K-nearest neighbor
RF	Random forest
SVR	Support vector regression
MLP	Multi-layer perceptron
GB	Gradient Boosting
DT	Decision tree
SOCF	Stacked OCF
UAW	Unadjusted actor weight
UUCW	Unadjusted use case weight
GS	Grid search
MAE	Mean Absolute Error
MMRE	Mean magnitude of relative error
MBRE	Mean balance relative error
MIBRE	Inverted balance relative error
MdMRE	Median magnitude of relative error
RMSE	Root mean square error

SSE	Sum of squares errors
PRED(x)	Percentage of prediction within x%
SA	Standardized accuracy
SLOC	Source lines of code
FPA	Function points analysis
COCOMO	Constructive cost model
SLIM	Software life cycle management
AOM	Algorithmic Optimisation Method
LOOCV	Leave on out cross-validation
OCF(PF_{CFE})	Effective productivity factor calculations

1 INTRODUCTION

1.1 Motivation

Software Project Development has evolved into a dynamic and competitive industry requiring high-level human resources. Software products are becoming more complicated, unpredictable, and challenging to control. Many research projects in the software field have been conducted in recent decades with the goal of steering software development processes into more regulated, manageable, and predictable paths. Project managers must estimate the cost of the software product as well as the resources, effort, and time required to complete a project on time and within budget [1]. Software measurement problems, such as project duration prediction or defect density, receive special attention. These issues demonstrate that the project management role has significantly increased.

Software Development Effort Estimation (SDEE) is critical to the overall success of solution delivery. Early SDEE in the first phase of the software development lifecycle is essential to avoiding project failures. The project manager's role is to look at software products to help with budgeting, scheduling, planning, project bidding, human resource allocation, and risk mitigation. The SDEE is vital for some reasons [2]. First, it is beneficial to make informed decisions about resource management before the project begins. The project plan is then used to make informed decisions about managing and planning the project. It is critical to allocate appropriate effort to the various activities in managing project development. As a result, this has led many researchers to study software estimation to obtain a more accurate SDEE [3], [4], [5]. However, based on the requirement specifications, the SDEE cannot be expected to produce correct results [6]. The issue of accurate effort estimation remains unresolved. An effort estimation method is used to reduce the risk of surprises during the project to the lowest possible value. It provides project managers with good control decisions to ensure that reasonable effort is allocated to the various activities throughout the project's development life cycle. When inaccurate models are used, such estimation decisions can have disastrous consequences. The most visible example of problems in managing complex, distributed software systems is the failure of many software projects [7]. The results show that actual effort and schedule are exceeded for most projects compared to estimates. If the software cost is underestimated, the project will be inefficient, and the actual price will undoubtedly be surpassed. Finally, even if completed on time, these overestimated projects usually become more extensive and costly than planned. In contrast, the functionality and quality of these underestimated projects are reduced to meet the plan's requirements. This can result in losing the bid or wasting time, money, personnel, and other resources, resulting in financial loss or even bankruptcy.

Use Cases can be helpful to measure the estimated effort at an early stage of a software project before the essential information is obtained during the requirements phase of the software lifecycle [8]. Neil et al. [9] surveyed the techniques used in the requirements elicitation, description, and modeling phases and found that the use cases were used in the early stages by more than half of the software projects. This has sparked the interest of numerous researchers in using use cases-based SDEE approaches and their initial applicability for greater accuracy. Karner [10] introduced the Use Case Points (UCP) method as a metric for sizing object-oriented software projects based on a structured scenario and actor analysis of the Use Case Model (UCM). Most studies focus on evaluating UCP as a potential early SDEE method that could be used to estimate software development effort and show its suitability for the software industry [11], [12], [13], [14].

Based on the literature reviewed above, this thesis focuses on developing SDEE methods for estimating software size and effort from UCM. Our methods can be used during the requirements phase of the software lifecycle. We aim to develop methods to handle imprecision and incorporate expert opinions to produce accurate and reliable effort estimates. With this objective, the thesis analyzes and proposes SDEE methods to reduce the impact of human error in UCM analysis and simplify the original principles of UCP. Each approach has its advantages, and they complement each other to form a complete process and promote significant efficiency to minimize the estimation error more efficiently in all situations. The results show that the proposed SDEE methods based on use cases are competitive with other alternatives.

1.2 Problem statement

UCP is a promising method for effort estimation in the early stages of software development that offers numerous benefits to the software industry [15], [16], [17]. Using machine learning to build SDEE models based on the original UCP formula could be a solution to improve its accuracy. Some approaches [18], [19], [20], [21], [22], [23] have also addressed variant models, especially regression models, to improve estimation accuracy based on historical data. The main drawback of the methods described above is that none of them is comprehensive or provides better accuracy in estimating software effort in all situations. There are still known problems in using UCP methods.

- The first problem is a particular uncertainty in evaluating technical complexity factors (TCF) and environmental complexity factors (ECF), as it depends on the experience of experts [24], [25], [26], [27], [28]. In particular, assigning an appropriate value to an ECF is difficult due to the lack of relevant information. This is because an ECF is associated with the level of information and experience of a particular software development team. Similar problems exist

in assigning a value to a TCF. These correction factors affect the estimation accuracy of UCP, so they need to be refined [29], [30]. Therefore, we will examine the close relationship between technical and environmental factors and prediction error to identify the best factors that significantly affect the estimation accuracy of the UCP method. This issue will be discussed in Chapter 4.1, as we have proposed a new formula for calculating the correction factors in the UCP method.

- The second problem is that potentially unsuitable variables are not considered in the UCP equation. In particular, use cases are written in natural language, and there is no rigorous process for assessing the quality or fragmentation of use cases. As a result, the number of steps in a use case may vary, affecting the estimate's accuracy. In addition, the estimate's accuracy may suffer if a use case contains multiple scenarios. Almost all previous methods for estimating software effort based on UCP have focused on developing the method by evaluating the complexity of the use case model and complexity weights [31], [32], [33], [34], [35], [36], [37]. However, we believe the regression approach based on UCP elements can solve this problem. Specifically, we will explore the implementation of multiple linear regression (MLR) models to select new formulas and regression coefficient values to reduce the impact of human error in evaluating actors or use cases. As shown in Chapter 4.2, this new formula outperforms the estimation accuracy of UCP.
- Moreover, given the complexity of today's software development projects, effort estimation requires the support of statistics and machine learning (ML). According to Kumar et al. [38], the overall estimation accuracies of SDEE methods based on statistical and machine learning techniques are almost acceptable as they are within 25% of the percent error (PRED (0.25)). The techniques are used to model the relationship between effort and software variables, which is particularly useful when the relationship is non-linear. However, one question is how to select unbiased approaches and appropriate algorithms. We note that single statistical and machine learning methods are unreliable, and the accuracy of a single method depends on its parameter configurations [39]. According to Thiago et al. [40], using a single model does not lead to optimal SDEE results. Priya et al. [41] also found that combining multiple models improves reliability. For all datasets, almost all ensemble SDEE approaches use the same learning parameter settings. With the above analysis, the thesis aims to reduce the bias and variability errors of the single models. In Chapter 4.3, we present the ensemble approach, which integrates seven well-known statistical and machine learning methods and fine-tunes the parameters of all the single methods to create a new and more comprehensive method in the early stages of software development.
- The fourth focus is the difficulty of converting software size into the corresponding effort. Many researchers consider the software productivity

factor, or the amount of software produced per effort, critical to estimating effort [42], [43], [44], [45]. This term also refers to the ratio of effort to size, also known as the productivity factor (PF). Most accepted values for the productivity factor have been suggested by project managers or use predetermined values for software productivity [46], [47]. However, we believe each software project takes place in a unique environment. Therefore, the question of whether to impose a fixed PF on all software projects has not been adequately addressed. This issue was discussed in Chapter 4.4 when we developed the software productivity model using the ensemble approach with historical correction factors. According to the findings, learning productivity values for each project is more useful and efficient than using predetermined values for all projects.

1.3 Research contributions

With each problem statement presented in section 1.2, we have made the following contributions:

- Regarding the first issue, we conducted several experiments on four datasets to identify the best technical and environmental complexity factors that significantly affect the estimation accuracy of the UCP method in the regression analysis. Our goal is to improve the estimation accuracy of the UCP method. We demonstrated an approach based on Least Absolute Shrinkage and Selection Operator (LASSO) method. Different values of the variable controlling the strength of the penalty are tested during the factor selection process to achieve the lowest prediction error. Finally, we propose a new formula for calculating the correction factors in the UCP method. This method is called Optimization Correction Factors (OCF).
- Continuing the development of the OCF method, we focused on modifying the OCF method to achieve more accurate estimates. Thus, we proposed an approach in which least square regression (LSR) or multiple linear regression (MLR) models are applied to the OCF-based elements to minimize estimation errors and the influence of unsystematic noise. An extended version of the OCF method called ExOCF (Extension of Optimizing Correction Factors) is proposed as a helpful method for project managers in the estimation phase.
- Seven different single models were proposed for estimating software size using OCF-based elements in the third objective. These statistical and machine learning models include K-Nearest Neighbor, Random Forest, Support Vector Regression, Multilayer Perception Gradient Boosting, Multiple Linear Regression, and Decision Tree. We analyzed the optimal parameter values for each technique to give it high predictive capacity. Based on this, we propose the OCF-based stacked generalization ensemble method of seven single

models, named Stacked OCF (SOCF). The study is primarily concerned with reducing individual model biases and variability errors. The proposed ensemble-based approach is a comprehensive approach to improve estimation accuracy and minimize project risks in the early stages of software development.

- Concerning the last objective presented in section 1.2, we conducted a project productivity model of the team developing a project. The overall productivity factor is based on correction factors in OCF method, which is built using an ensemble construction mechanism of three popular ML techniques, including Support Vector Regression, Multiple Linear Regression, and Decision Tree. We chose these techniques because they have fast learning time, good generalization ability, and a more straightforward design. A voting ensemble is an ensemble ML model to determine the relationship between project productivity and independent correction factor variables. In this way, our method can benefit from good research on correction factors in OCF and reduce the estimation error of the methods compared to using fixed productivity metrics.

1.4 Organization of the thesis

The remainder of the thesis is organized as follows: Chapter 2 defines the terms used in the thesis, such as the use case points method, statistical and ML techniques, and the measurement criteria to evaluate the estimation accuracy of the SDEE methods. Chapter 3 introduces a literature review of existing research related to SDEE, as well as some related work on UCP-based effort estimation and SDEE tools for the software industry. Chapter 4 presents four proposed approaches. The research methodology is presented in Chapter 5, which includes a description of the datasets for the experiment as well as the empirical procedure for evaluating the proposed methodologies. Chapter 6 presents the results and discussion of the proposed approaches. Threats of the validity of this study is presented in Chapter 7. The contribution of the thesis to science and practice is summarized in Chapter 8. Finally, Chapter 9 presents the conclusion and future works.

2 THEORETICAL FRAMEWORK

This chapter defines the terms used in the thesis, such as the use case points method, statistical and machine learning techniques, and the evaluation criteria.

2.1 Use Case Points method

The UCP method was first presented by Karner [10] for estimating the software size of object-oriented software projects in the early phases of software development. The method measures software size based on use case diagrams. In particular, the method provides a process for converting use case diagram elements into quantitative information that can be applied to four simple size metrics. The metrics of UCP are shown in Figure 2-1.

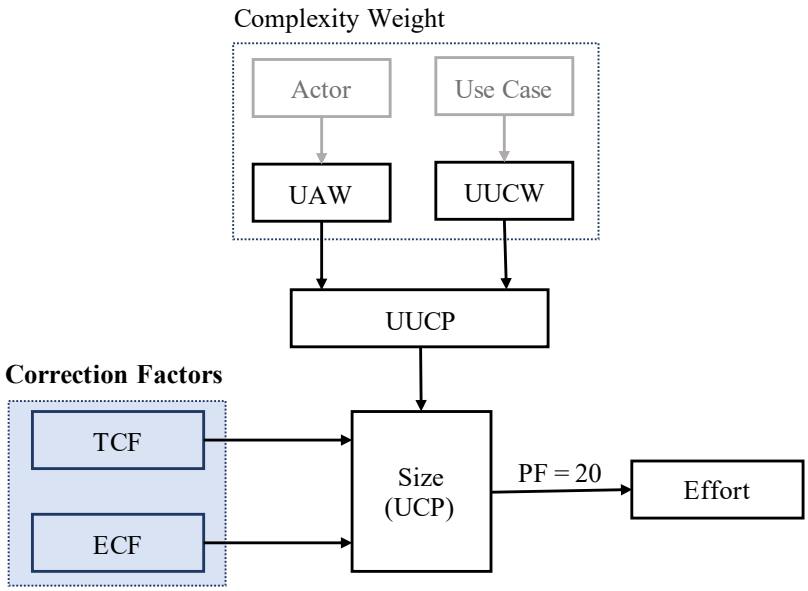


Figure 2-1. The process of the Use Case Points method

First, the unadjusted actor weight (UAW) is calculated as shown in Eq. (2.1). According to their complexity, actors are classified into three levels (simple, average, and complex) as described in Table 2-1. Specifically, simple actors describe the system through an API. Average actors represent the system through a protocol. Complex actors represent the system through a graphical user interface.

$$\mathrm{UAW} = \sum_{i=1}^{3} a_i \times w_i \tag{2.1}$$

where a_i is the number of the actor in the actor type, w_i is the complexity weight of actor type.

Second, the unadjusted use case weight (UUCW) is calculated as shown in Eq. (2.2). The use cases are classified into three levels (simple, average, and complex) based on the number of transactions in the use case description, as shown in Table 2-2. A simple use case includes less than 4 transactions, an average use case includes between 4 and 7 transactions, and a complex use case includes more than 7 transactions.

$$\mathrm{UUCW} = \sum_{j=1}^{3} u_j \times w_j \tag{2.2}$$

where u_j is the number of the use case in the use case type, w_j is the complexity weight of use case type.

Next, correction factors, i.e., Technical Complexity Factors (TCF) and Environmental Complexity Factors (ECF), are used to represent the experience level of the software development team. The Technical Complexity Factors in Eq. (2.3) are calculated from 13 factors. Each factor has a value between 0 and 5 with the corresponding weighting, as shown in Table 2-3.

$$\mathrm{TCF} = 0.6 + 0.01 \sum_{i=1}^{13} T_i \times Wt_i \tag{2.3}$$

where T_i is the value of technical complexity factor, Wt_i is the complexity weight of technical factor.

The Environmental Complexity Factors in Eq. (2.4) are calculated from 8 factors, each factor has a value between 0 and 5 with the corresponding weighting, as shown in Table 2-4.

$$\mathrm{ECF} = 1.4 - 0.03 \sum_{i=1}^{8} E_i \times We_i \tag{2.4}$$

where E_i is the value of the environmental complexity factor, and We_i is the complexity weight of the environmental factor.

The UCP is computed using Eq. (2.5) as follows:

$$\mathrm{UCP} = (\mathrm{UAW} + \mathrm{UUCW}) \times \mathrm{TCF} \times \mathrm{ECF} \tag{2.5}$$

Finally, the obtained UCP are multiplied by the PF to give the final effort. This is shown in Eq. (2.6).

$$\mathrm{Effort} = \mathrm{Size} \times \mathrm{PF} \tag{2.6}$$

where Effort is measured in person-hours and Size is measured in UCP. For SDEE, Karner proposed 20 person-hours to develop each UCP.

Table 2-1. Actor classification and their complexity weights

Actor classification	Description	Weight
Simple	The systems through an API	1
Average	The system through a protocol	2
Complex	The system through GUI	3

Table 2-2. Use Case classification and their complexity weights

Use case classification	Number of transactions	Weight
Simple	(0, 4)	1
Average	<4, 7>	2
Complex	(7, ∞)	3

Table 2-3. Technical complexity factors (TCF)

Factor	Description	Weight
T1	Distributed System	2
T2	Response Adjectives	2
T3	End-Use Efficiency	1
T4	Complex Processing	1
T5	Reusable Code	1
T6	Easy to install	0.5
T7	Easy to Use	0.5
T8	Portability	2
T9	Easy to Change	1
T10	Concurrency	1
T11	Security Features	1
T12	Access for Third Parties	1
T13	Special Training Facilities	1

Table 2-4. Environment complexity factors (ECF)

Factor	Description	Weight
E1	Family with RUP	1.5
E2	Application Experience	0.5
E3	Object-oriented Experience	1
E4	Lead Analyst Capability	0.5
E5	Motivation	1
E6	Stable Requirements	2
E7	Part-time Workers	-1
E8	Difficult Programming Language	2

2.2 Statistical and machine learning techniques

Given the complexity of today's software development projects, we discovered that estimating effort without the assistance of statistical and machine learning models is impossible. We summarised several selected studies [48], [49], [50], [51], [52], [53], [54], [55], [56], [57], [58], [59], [60], [61], [62], [63] on estimating software effort with statistical and machine learning models from 2015. Based on the above review, in this section, we define the seven most commonly used statistical and ML algorithms for SDEE and their configuration parameters in this work. The accuracy of a given statistical or machine learning method is determined by the configuration parameters that describe the characteristics of a given dataset. Choosing the optimal parameter values for a technique gives it a high predictive capacity. Grid Search (GS) [64] is used to optimize the configuration parameters of each statistical and ML technique. Specifically, GS searches the parameter set of each empirical method in a predefined range of values for each dataset and then selects the configuration that leads to the "optimal" estimates. The parameter search ranges were derived from previous analyses [54], [59]. We extended the search range in each case to include as many possible configurations.

These models include Multilayer Perception, Support Vector Regression, Decision Tree, Random Forest, K-Nearest Neighbour, Gradient Boosting, and Linear Regression. These models are learned and trained using historical project data and can be applied in two scenarios. The models can stand alone by taking inputs such as software size and productivity and producing outputs such as software effort. The models can also calibrate some parameters or weights of algorithmic models.

2.2.1 Multilayer perceptron

The multilayer perceptron (MLP) is a feedforward neural network typically trained to solve regression problems using a backpropagation algorithm. The simplest MLP model has three nodes: an input layer, a hidden layer, and an output layer [65]. The input layer has the same number of nodes as the independent variables identified in the input pattern. Each neuron in the hidden layer uses a nonlinear activation function to convert the values from the preceding layer using a weighted linear summation. The output layer's nodes are defined by the problem and the number of dependent variables.

One of the most essential steps in the development of the MLP is the optimization of its configuration parameters, such as the number of neurons in the hidden layer and the three parameters of the learning algorithm (initial learning rate, momentum, and regularization term). Linoff et al. [66] recommend that the number of nodes in the hidden layer should be between the number of nodes in the input layer and twice this number. The critical parameters for constructing the MLP model and their values for preliminary execution are depicted in Table 2-5.

Table 2-5. The parameters for constructing the MLP model.

Model parameter	Search range
Initial learning rate	$L = \{0.01, 0.02, 0.03, 0.04, 0.05\}$
Number of hidden nodes	$H = \{5, 6, 7, 8\}$
Momentum	$M = \{0.1, 0.2, 0.3, 0.4, 0.5\}$
Regularization term	$\alpha = \{0.00001, 0.0001, 0.001, 0.01\}$

2.2.2 Support vector regression

Support Vector Machine (SVM) is a supervised learning method based on statistical learning theory [67]. SVR is a special form of SVM used to model the input-output functional relationship or regression. Assume that the training dataset $D = \{(x_i, y_i)\}_1^n$ where $x_i \in \mathbb{R}^m$ denotes the input values, $y_i \in \mathbb{R}$ denotes the corresponding output values, n is the number of samples in the training dataset, and m is the dimension of input dataset.

The goal of SVR is to approximate the nonlinear relationship shown in Eq. (2.7) that $f(x_i)$ is as close as possible to the obtained target value (y_i).

$$y_i = f(x_i) = \langle w, \Phi(x_i) \rangle + b \tag{2.7}$$

where $w \in \mathbb{R}^m$, $b \in \mathbb{R}$ are respectively the weight vector and threshold, $\langle ., . \rangle$ denotes dot product, and $\Phi(x_i)$ is the transformation function which maps the input values from $\mathbb{R}^m$ space to a feature space with higher dimension. The values

w and b are reduced to ensure that the approximated function satisfies the above objective.

$$\min_{w,\xi,\xi^*} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \xi_i + \xi_i^* \tag{2.8}$$

subject to

$$\begin{gathered} y_i - \langle w, \Phi(x_i)\rangle - b \le \varepsilon + \xi, i = 1, \dots, n \\ \langle w, \Phi(x_i)\rangle + b - y_i \le \varepsilon + \xi^*, i = 1, \dots, n \\ \xi \ge 0, i = 1, \dots, n \\ \xi^* \ge 0, i = 1, \dots, n \end{gathered} \tag{2.9}$$

where ε is a deviation of a function $f(x_i)$, ξ and ξ^* are slack variables used to measure ε. The regularization parameter C defines the error tolerance over ε.

$\varepsilon-$SVR is used as a variant of SVR, and the Radial Basis Function (RBF) is usually used as a kernel function. The RBF kernel is calculated as

$$K(x_i, x_j) = \exp\left(-\gamma\|x_i - x_j\|^2\right), \gamma > 0 \tag{2.10}$$

Three parameters that significantly affect the performance of the $\varepsilon-$SVR generalization, namely, the C, ε, and γ parameter, must be carefully selected. Table 2-6 shows the details of these configuration parameters and their search ranges for the SVR method.

Table 2-6. The parameters for constructing the SVR model

Model parameter	Search range
Regularization term	$C = \{5, 10, 100, 150\}$
Epsilon for $\varepsilon-$SVR	$\varepsilon = \{1, 0.1, 0.01, 0.001, 0.0001\}$

2.2.3 Decision tree

Decision trees (DT) are supervised machine learning to solve regression and classification problems [68]. A decision tree creates a flowchart in an inverted tree-like structure where the internal nodes illustrate the test, the branches define the test result, and each leaf node denotes a class label [69]. The output of a given DT is partitioned into distinguished leaf nodes, following certain conditions such as an if/else loop. DTs have many variants of algorithms such as ID3, CART, CHAID, C4.5, M5P, and REPTrees [70]. The DTs used in this study are an optimized version of the CART algorithm.

In the DT, four parameters need to consider: (1) The maximum depth (max_depth) - If it is not specified, the tree is expanded until the last leaf nodes

contain a single value, resulting in overfitting. (2) The minimum number of leaf nodes (min_samples_leaf) in a decision tree to control the complexity of the model. (3) The minimum weighted fraction of the sum total of weights (min_weight_fraction_leaf) required to be at a leaf node. (4) The number of leaf nodes (max_leaf_nodes) to control overfitting. Too high values may also lead to under-fitting. Table 2-7 provides details concerning the parameters for the DT model and their search ranges.

Table 2-7. The parameters for constructing the DT model

Model parameter	Search range
The maximum depth of the tree	$\max\_depth = \{3, 5, 7, 9, 11, 12\}$
The minimum weighted fraction	$\text{min\_weight\_fraction\_leaf} = \{0.1, 0.2, 0.3, 0.4, 0.5\}$
The number of leaf nodes	$\text{max\_leaf\_nodes} = \{10, 20, 30, 40, 50, 60, 70, 80, 90\}$
The minimum number of samples	$\text{min\_sample\_nodes} = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$

2.2.4 Random forest

The random forest technique (RF) uses the supervised nonparametric approach for regression and classification [71], [72]. It creates multiple DTs and combines them to obtain a more accurate and stable prediction. The result of RF is the maximum vote of a panel of independent judges, which makes the final prediction better than the best judge. The parameters for developing an RF model must be considered to increase predictive power and facilitate model training. These parameters determine whether the predictions are more robust and stable [63]. The parameters used in building an RF model are as in the DT model (see Table 2-8).

Table 2-8. The parameters for constructing the RF model

Model parameter	Search range
The number of trees	$\text{n\_estimators} = \{100, 150, 200, 150, 300, 350, 400, 450\}$
The minimum number of samples	$\text{min\_sample\_nodes} = \{1, 2, 4\}$
The maximum depth of the tree	$\text{max\_depth} = \{10, 20, 30, 40, 50, 60, 70, 80, 90, 100\}$

2.2.5 K-Nearest neighbors

K-nearest neighbors (KNN) is a non-parametric machine learning method used in classifications and regressions. KNN collects historical data, called the training dataset, and produces estimates for new test data. The k-nearest data from the training data set is determined. Based on the data attributes of the closest data sets, an estimate is made for the new data. In KNN, the selection of K (number of neighbors) is very crucial. The algorithm becomes sensitive to noise if the K value is too small. If the K value is too large, datasets of other classes can be counted as nearest neighbors [73].

Table 2-9 shows the values of its search range. The default Euclidean distance in Scikit-learn to measure the distance between points in KNN.

The Euclidean distance $d(p_i, q_i)$ between one vector $p = (p_1, p_2, \dots, p_n)$ and another vector $q = (q_1, q_2, \dots, q_n)$ can be computed as follows:

$$d(p_i, q_i) = \left[\sum_{i=1}^{n} (p_i - q_i)^2\right]^{1/2} \tag{2.11}$$

Table 2-9. The parameter for constructing the KNN model

Model parameter	Search range
Number of neighbours	$K = \{2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15\}$

2.2.6 Gradient boosting

Gradient Boosting (GB) is a ML technique used in regression and classification tasks. GB is based on the idea of an ensemble method derived from a decision tree [74]. In GB, the choice of the number of decision trees (number of estimators) is a crucial parameter. The more the number of decision trees, the better the data learning. However, adding many trees can significantly slow down the training

process. Therefore, a parameter search is necessary. Three other parameters of interest in GB are the number of boosting stages (n_estimators), the minimum number of leaf nodes (min_samples_leaf), and the maximum depth of the single regression estimators (max_depth), which is used to control model overfitting. Table 2-10 shows the search ranges of the configuration parameters of the GB model.

Table 2-10. The parameters for constructing the GB model

Model parameter	Search range
Number of boosting stages	$\text{n\_estimators} = \{20, 40, 60, 80, 100\}$
Minimum number of leaf nodes	$\text{min\_samples\_leaf} = \{10, 20, 30, 40, 50, 60, 70\}$
Maximum depth	$\text{max\_depth} = \{5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16\}$

2.3 Evaluation criteria

In the SDEE, several criteria are required to evaluate the estimation accuracy of the methods. Table 2-11 summarizes the accuracy measures used in estimation studies related to our work (2016 onward). The Mean Magnitude of Relative Error (MMRE) is the most commonly used standard for SDEE accuracy [1], [75]. However, this metric is prone to bias [21], [22]. For this reason, we use five alternative criteria to achieve a fair and symmetric distribution. These include Mean Absolute Error (MAE), Mean Balance Relative Error (MBRE), Mean Inverted Balance Relative Error (MIBRE), Median Magnitude of Relative Error (MdMRE), and Root Mean Square Error (RMSE). We also used two measures to assess the accuracy of the estimation models: the Sum of Squares Errors (SSE) and the Percentage of prediction within x% (PRED (x)). Two measures are critical for estimating the variation in modeling error [19]. They are chosen because of their ability to describe errors in specific datasets. PRED(x) is less biased to underestimation and generally determines the same best method as Standardized Accuracy (SA). An SDEE method with high estimation accuracy (when PRED (x) values are high) is also reasonable (when SA values are high) [76].

Table 2-11. Summary of the accuracy measures used in SDEE methods

Cited study	MMRE	PRED	MBRE	MIBRE	MAE	SA	MAPE	MSE	RMSE	NRMSE	SSE	R^2	RSS
[77]			x	x	x	x							
[78]							x	x	x	x	x		
[62]								x	x			x	x
[55]			x	x	x	x							
[79]			x	x	x	x							
[80]			x	x	x	x							
[81]			x	x	x	x							
[82]	x												
[63]			x	x	x								
[83]	x	x											
[84]					x				x				
[85]								x	x		x		
[86]	x												
[87]	x	x											
[14]	x	x									x	x	
[88]	x	x					x	x			x	x	
[89]			x	x	x	x							

- Mean Absolute Error (MAE)

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i| \tag{2.12}$$

- Mean Balance Relative Error (MBRE)

$$\mathrm{MBRE} = \frac{1}{n}\sum_{i=1}^{n} \frac{|(y_i - \hat{y}_{i)}|}{\min(y_i - \hat{y}_i)} \tag{2.13}$$

- Mean Inverted Balance Relative Error (MIBRE)

$$\mathrm{MIBRE} = \frac{1}{n}\sum_{i=1}^{n}\frac{|(y_i - \hat{y}_{i)}|}{\max(y_i - \hat{y}_i)} \tag{2.14}$$

- Median of Magnitude of Relative Error (MdMRE)

$$\mathrm{MdMRE} = \mathrm{median}_i\left(\frac{|y_i - \hat{y}_i|}{y_i}\right) \tag{2.15}$$

- Root Mean Square Error (RMSE)

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}} \tag{2.16}$$

- Sum of Squares Errors (SSE)

$$\mathrm{SSE} = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2.17}$$

- Percentage of Prediction within x% (PRED(x))

$$\mathrm{PRED}(x) = \frac{1}{n}\sum_{i=1}^{n}\begin{cases}1 & \text{if } \dfrac{|y_i - \hat{y}_i|}{y_i} \le x \\ 0 & \text{otherwise}\end{cases} \tag{2.18}$$

where n is the number of observations, y_i is the real known value, $\hat{y}_i$ is the predicted value, and $x = 0.25$.

3 CURRENT STATE OF THE ISSUES DEALT WITH

This chapter presents a literature review of existing research related to SDEE, some related work on UCP-based effort estimation, and SDEE tools for the software industry.

3.1 Existing research related to SDEE

Existing SDEE research can be divided into three categories [6], [90], [91]. These include algorithmic, non-algorithmic, and estimation by statistical and machine learning models.

3.1.1 Algorithmic effort estimation models

Algorithmic models include Source Lines of Code (SLOC) [1], Function Point Analysis (FPA) [92], [93], Constructive Cost Model (COCOMO) [1], Use Case Points (UCP) [10], and Software Life Cycle Management (SLIM) [94]. These models, which are still the most widely used in the literature [95], [96], use mathematical equations to estimate the cost of a software project. Table 3-1 below describes the algorithmic models.

Table 3-1. Algorithmic models

Estimation method	Description
Source Lines of Code (SLOC)	The method uses the number of source lines developed to measure the software size. However, the size of a change request cannot be accurately estimated using the SLOC method until the coding process is completed. As a result, estimating the size of a change request in the early stages of software development is nearly impossible [97]. Because software size is a critical input to an effort model, an incorrect SLOC estimate will result in an inaccurate effort estimate.
Function Point Analysis (FPA)	Based on the functions provided to the end user, the method estimates the complexity and size of a software system. There are several methods for counting function points. Nevertheless, the method administered by Function Points Analysis (FPA), based on the International Function Point Users Group (IFPUG) [98], is the standard method. Internal Logical File (ILF), External Interface File (EIF), External Input (EI), External Output (EO),

	and External Inquiry (EQ) are five parameters to measure the software size.
Constructive Cost Model (COCOMO)	COCOMO 81 is a software cost prediction method developed in 1981 [1] that uses a simple regression formula. The model's parameters are derived from historical projects and current project characteristics. The COCOMO II [2] model is an enhanced version of the COCOMO 81 better suited for project estimation in modern software development. Specifically, COCOMO II employs logical SLOC, whereas COCOMO 81 employs physical SLOC. Multiple physical SLOCs can be contained within a logical SLOC (if-then-else).
Use Case Points (UCP)	The method is based on the elements of the system use cases with technical and environmental aspects. The method is based on a calculation with four elements: Unadjusted Use Case Weight (UUCW), Unadjusted Actor Weight (UAW), Technical Complexity Factor (TCF), and Environmental Complexity Factor (ECF).
Software Life Cycle Management (SLIM)	The Putnam model is another name for this model. The SLIM describes the effort and time required to complete a project of a specific size using the Norden/Rayleigh function. The model can save analysis data from previous projects, which can then be used to calibrate and build the workforce in an existing dataset by answering a series of questions.

3.1.2 Non-algorithmic effort estimation models

Expert judgement [99], Analogue-based [100], Price-to-win [101], Top-down [101], Bottom-up [101], Wideband Delphi [1], and Planning Poker [102] are examples of non-algorithmic models. These models estimate software development costs by drawing on an expert's prior experience or historical projects. Table 3-2 contains descriptions of the non-algorithmic models.

Table 3-2. Non-algorithmic models

Estimation method	Description
Expert judgement	The method is based on the consultation of an expert's experience, knowledge, motivation, and field knowledge and the exchange of analysts and experts to propose an estimate for a specific project. Delphi technique is used to facilitate communication and collaboration among experts. The primary advantage is that it can be accurately compared with other models if the experts know exactly the problem area of the proposed project.
Analogy-based	An analogy-based method is a systematic form of expert judgment. It is an application of the Cased Based Reasoning method. The method compares similar historical projects, documenting all necessary information.
Bottom-Up and Top-down Approach, Price-to-win	The methods are entirely based on software project budgets, either broken down by project module (top-down) or predicted as the sum of project module estimates (bottom-up).
Wideband Delphi	The method is a team-based method for estimating software costs based on team agreements. The work breakdown structure (WBS) is used to estimate costs.
Planning Poker	The method is also a consensus-based estimation method like Wideband Delphi. The method is used in agile software development for cost estimation and is consistent with agile software development's people-oriented approach.

3.1.3 Estimation model by statistical and ML models

Statistical and ML models allow estimation using information from previously completed projects. Utilizing this learning mechanism, experts spend less time on project estimation and more time on other software system functions that satisfy the customer. Over the past decade, researchers have focused on using statistical and machine learning techniques. According to their findings, the estimation accuracy obtained with ML techniques was significantly higher than that obtained with non ML-based estimation methods [103]. The overall estimation accuracy of SDEE methods based on statistical and machine learning techniques is almost

acceptable and within 25% of the percent error (PRED (0.25)) [38]. According to our findings [48], [49], [50], [51], [52], [53], [54], [55], [56], [57], [58], [59], [60], [61], [62], [63], seven statistical and ML techniques, namely MLP, SVR, DT, RF, MLR, KNN, and GB, were the most commonly used in SDEE (see Figure 3-1).

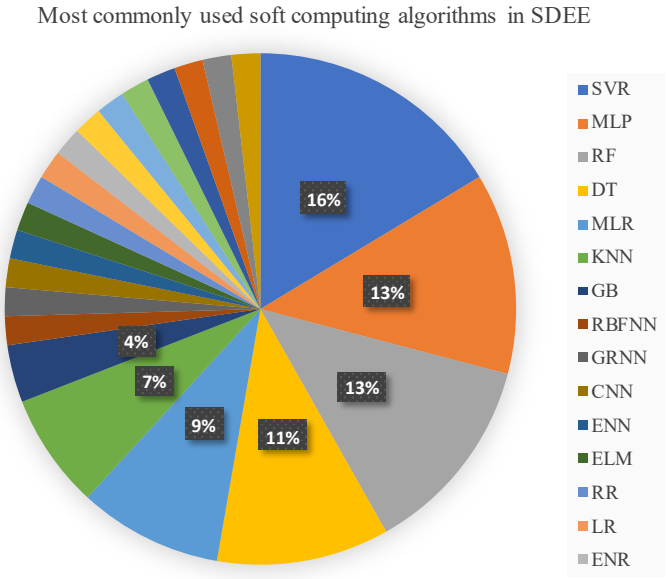


Figure 3-1. Most commonly used statistical and ML algorithms in SDEE

3.2 Related work for UCP-based effort estimation

We discussed some problems with the UCP model. One approach is to focus on increasing the complexity levels for use case weighting, actor weighting, or both, discretizing the existing complexity levels, and calibrating the complexity weights. For example, in Re-UCP, Manzoor et al. [15] added actor weighting and use case weighting, and in the UCP sizing method, the authors added another level of evaluation to the use case weighting system. Nunes et al. [24] gave six new actor weights. Wang et al. [16] extended the complexity levels of use cases by incorporating fuzzy set theory and Bayesian belief networks into the UCP model. In the e-UCP method, Periyasamy et al. [17] changed the complexity levels of actors and reclassified the complexity of use cases. The UCPabc approach [31] applies an activity-based costing method to all variables in the UCP method, and the productivity factor was changed to 8.2 person-hours. For incremental development estimates in large projects, an adaptation approach to the UCP method known as Adapted UCP (AUCP) [32] is used. Braz et al. [33] have proposed two methods for calibrating the internal level of the use case: use case size points (USP) and fuzzy use case size points (FUSP). A USP model introduces new components into the structure of a use case, namely the number and weights of scenarios, actors, preconditions, and postconditions. A FUSP model is a more

complex version of a USP that uses fuzzy set theory to solve some use case classification problems. Qi et al. [34] improved the accuracy of UCP estimation by calibrating the case complexity weights using Bayesian analysis. Rak et al. [35] proposed the use case reusability (UCR) model for estimating effort by assigning a new classification to use cases.

Regarding SDEE methods based on statistics and ML techniques, we divided them into three groups. The first group uses individual techniques to develop new methods based on the original method or to validate existing methods in industrial applications, focusing on improving accuracy. A UCP-based effort estimation model based on fuzzy logic and neural networks [30], a regression model and the Sugeno Fuzzy Inference System (SFIS) approach [104], or Cascade Correlation Neural Network (CCNN) model [105] are introduced to improve the estimation accuracy. The results show that these models improve the accuracy by 11%. The Adaptive Neuro-Fuzzy Use Case Size Point (ANFUSP) model is presented to estimate the effort required for object-oriented software projects [106]. Compared to the UCP method, the model's results are more accurate. A hybrid model based on Analogy-Based Estimation (ABE) and the Particle Swarm Optimization (PSO) algorithm is proposed to construct an attribute system with different weights depending on the cluster [20]. The results of the model revealed noticeably improved estimate accuracy.

The second group applies regression models by analyzing the validity of UCP variables. For example, the regression model accounts for the nonlinear relationship between software size and effort in person-hours [107]. Jorgensen [18] reported all variables included in the models to illustrate the accuracy and bias variation of the SDEE methods using regression analysis. Ochodek et al. [108] simplified the UCP method by discarding the UAW, measuring the UCP based on steps, or calculating the total number of steps in use cases. The Algorithmic Optimisation Method (AOM) is proposed to increase the accuracy of the correction coefficients of the effort estimation process [23] by employing multiple least squares regression with all UCP elements. The authors then conducted several experiments on two different datasets to investigate the significance of the UCP variables [62].

The last group employs ensemble methods, which produce more accurate results than single statistical and machine learning methods [103]. These studies concentrated on aspects of effort estimation such as base model diversity, model ranking within the ensemble, aggregation techniques, and model selection. Many researchers concurred that ensemble models outperformed single methods. In particular, an AdaBoost ensemble approach based on two single methods (KNN and SVR) [13], another ensemble of three single methods (SVR, MLP, and GLM) [109], an ensemble model of optimal trees (RF and RT) [110], an ensemble model combining UCP, expert judgment, and Case-Based Reasoning (CBR) [111], an ensemble framework based on an extended RF algorithm [112] are proposed. The

findings demonstrate that these ensemble methods perform better than single models and produce more accurate estimates of error measurements.

Although previous works have been remarkably well handled, selecting unbiased approaches and appropriate algorithms has proven difficult [113], [114]. It is well known that a method's accuracy depends on its parameters' settings [39]. However, most ensemble SDEE methods use the same learning parameter settings for all datasets. None of the above studies is exhaustive or guarantees higher accuracy in estimating software effort under all circumstances. Moreover, few studies have validated their results with statistical tests. In [115], it is argued that it is invalid to claim that one model is superior to another if adequate statistical tests are not performed.

3.3 Software estimation tools in the software industry

In this chapter, we aim to introduce the Use Case Estimation, a function in Enterprise Architect that gives a starting point for predicting project effort. It can be found at https://sparxsystems.com/products/ea/16.0/index.html. Figure 3-2 illustrates the screen of the Use Case estimation tool.

Use Case Estimation is a comprehensive project estimation tool that uses Use Case and Actor factors to determine effort. The tool can provide a reasonable estimate of the complexity of a system. The complexity of the work environment is determined by a set of weighted technical and environmental complexity factors, and use cases and actors are classified as Easy, medium, or complex. Based on Karner's UCP approach, the method produces a metrics report that includes the project estimation analysis and can be included in the project documentation.

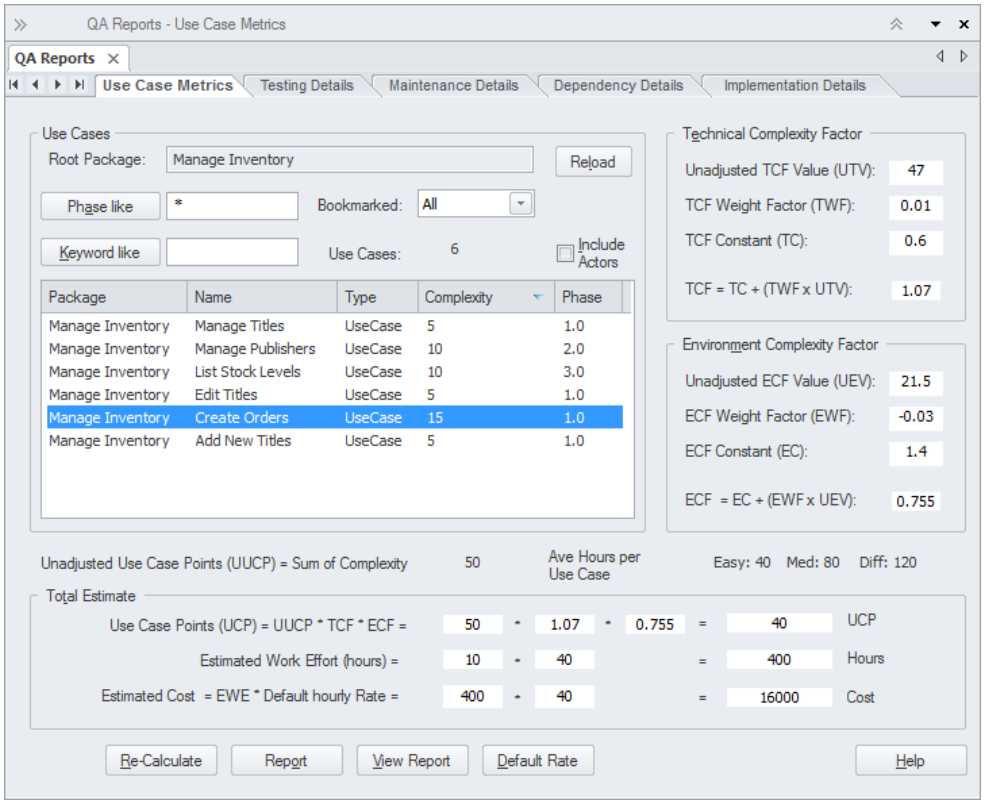


Figure 3-2. Use Case Estimation tool

An option that relies on manual transcripts or spreadsheets is Manuscript (https://manuscript.com), which is more of an issue tracker/project planner with built-in statistical modeling capabilities in the software. In addition, business.com has identified the top three software solutions for cost estimating in 2022. (updated August 26, 2022). CoConstruct (https://www.coconstruct.com/), MeasureSquare (https://www.measuresquare.com/) and STACK (https://www.stackct.com/) are three of these tools. These tools can estimate bid prices and other project parameters leading to a construction contract. However, depending on the particular company's needs, each option has advantages and disadvantages. The project manager is in charge of choosing the best estimating software. They must evaluate the strengths and limitations of project management to choose the best solution. Understanding the basics of estimating software solutions can help project managers narrow their options based on key features such as ease of use, pricing, stability, and customer support.

4 THE PROPOSED METHODS

The proposed methods are presented in this chapter based on the discussion of the previous chapters. First, a proposal for increasing the estimation accuracy of the existing UCP method was called Optimization Correction Factors (OCF) [116]. We analyze correction factors to identify the best technical and environmental complexity factors that significantly affect the estimation accuracy of the UCP method in regression analysis. To put this idea into practice, we propose a new formula to calculate the correction factors in the UCP method. Then, to obtain more accurate estimates, we aim to apply the MLR models to improve the ability of the OCF method to estimate the software size and minimize the prediction error. This is referred to as the Extension of Optimizing Correction Factors (ExOCF) [117]. The OCF variables are used in this method to determine the software size. The MLR formulation was created to estimate the software size values. Following the proposed ExOCF is another alternative framework for effort prediction to improve the overall performance of the regression. A novel Stacked SVR-MLR-MLP-DT-RF-KNN-GB on the OCF (SOCF) model is proposed to improve the overall performance of the regression. The model includes seven statistical and ML techniques: MLR, KNN, SVR, MLP, RF, GB, and DT. Finally, the calculations of the effective productivity factor (PF_{CFE}) are proposed in conjunction with the OCF as predictors of effort [118]. The summary of the four proposed methods is shown in Figure 4-1.

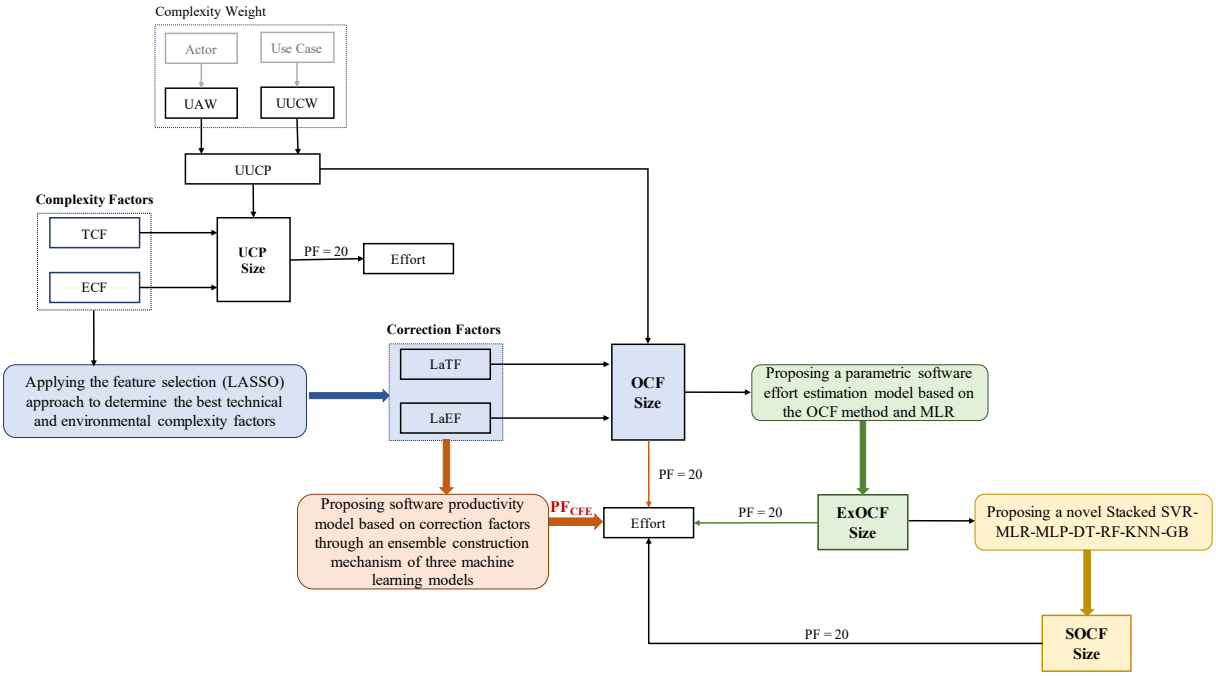


Figure 4-1. The proposed methods

4.1 The proposed Optimization Correction Factors method

4.1.1 Least absolute shrinkage and selection operator

The Least Absolute Shrinkage and Selection Operator (LASSO) is used to address variable selection in regression analysis [119]. LASSO regression is a method that performs two main tasks: L1 - regularisation and feature selection. It forms a constraint on the sum of the absolute values of the model variables, where the sum is required to be less than an upper bound (a fixed value). The method applies a shrinking - (regularisation) process, which penalizes regression variables - (correction value) coefficients by shrinking some of them to zero.

During the feature selection process, the correction values that still have a non-zero coefficient after the shrinking process are selected to form part of the model. This process aims to minimize prediction error - (the sum of squared errors - with an upper bound on the sum of the absolute values of the model parameters). The LASSO method is defined by the solution to the l_1 optimisation problem - (the formulation used by Buhlmann et al. [120]:

$$\text{minimize } \left(\frac{\|Y-X\beta\|_2^2}{n}\right) \qquad \text{subject to } \sum_{j=1}^{k} \| \beta \|_1 < t \tag{4.1}$$

where, t is the upper bound for the sum of the coefficients. This optimisation problem is equivalent to the parameter estimation below:

$$\hat{\beta}(\lambda) = \underset{\beta}{\operatorname{argmin}} \left(\frac{\| Y - X\beta \|_2^2}{n} + \lambda \| \beta \|_1 \right) \tag{4.2}$$

where, $\| Y - X\beta \|_2^2 = \sum_{i=0}^{n} (Y_i - (X\beta)_i)^2$, $\| \beta \|_1 = \sum_{j=1}^{k} |\beta_j|$ and $\lambda \geq 0$ is the variable that controls the strength of the penalty; the larger the value of λ, the greater the amount of shrinkage. The relation between λ value and fixed value t is a reverse relationship. It is certain that, as t becomes infinity, the problem becomes an ordinary least square; and λ will become 0. However, this is vice versa when t reaches 0, all coefficients will shrink to 0, and λ will go to infinity.

In the OCF, we use LASSO regression for its variable selection properties (Figure 4-2). When we minimise the optimisation problem, some coefficients are shrunk to zero - i.e. $\hat{\beta}_i(\lambda) = 0$; for some values of j - (depending on the value of parameter λ). In this way, features with coefficients equal to zero are excluded from the model.

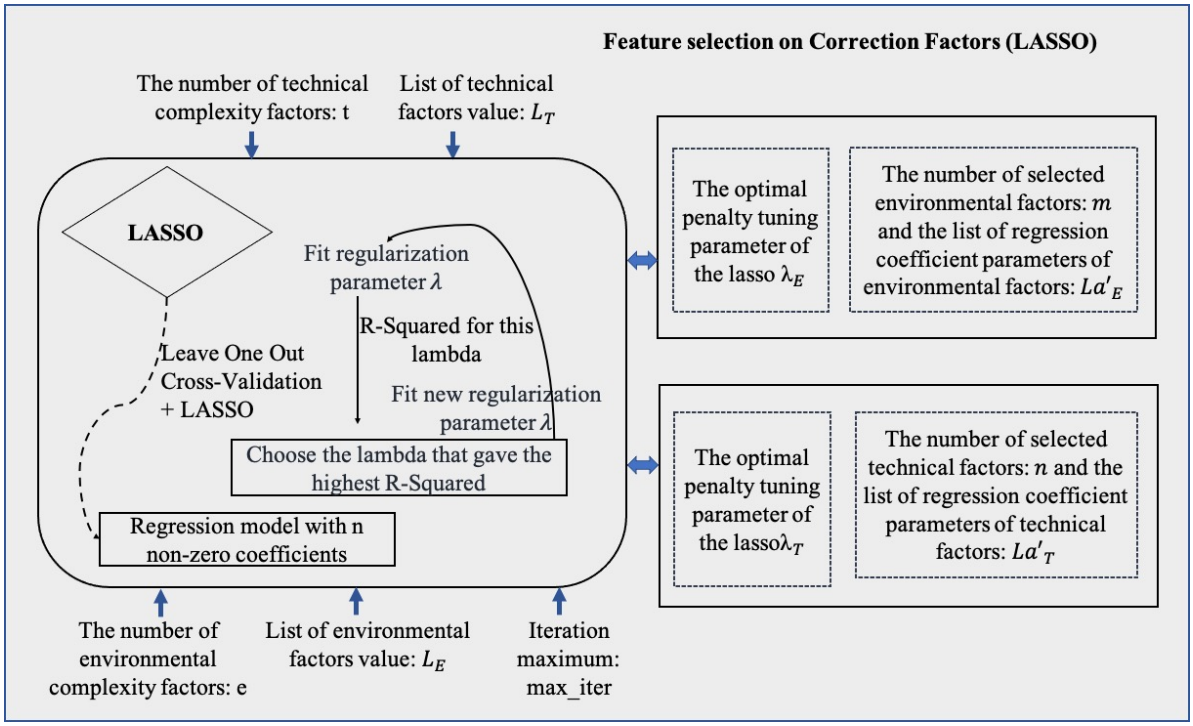


Figure 4-2. The detailed illustration of the feature selection on correction factors

4.1.2 Correction factors analysis

The evaluation of the correction factors depends on the experience of experts, which has a certain degree of uncertainty [81]. It is difficult to assign an appropriate value to an ECF because of a lack of relevant information. The reason is that an ECF is linked to the level of information and experience of a particular software development team. Therefore, it is difficult to suggest that the team evaluate their work - especially when the project managers are estimators and have to assign values to the ECF [121]. There are similar problems with the value assigned for a TCF. For example, factor T10 (Concurrent) shows a certain difficulty level. This factor could include parallel processing, parallel programming - or whether the system is stand-alone or interfaces with several other applications. The assignment of values to this factor may not be accurate, as there are no guidelines in the UCP that explain this factor precisely. Huanca et al. [29] identified the main factors affecting the accuracy of the estimation of the UCP method - which is ECF and TCF. According to this review, these factors affect the estimation accuracy and require a re-evaluation. A slight variation in the weight value of the adjusting factors could dramatically affect the software size and then the estimating effort. Nassif et al. [30] also highlighted the need to refine the parameters as adjusting factors directly related to estimations calculated using the UCP method.

4.1.3 Optimizing Correction Factors method

A detailed illustration of the OCF method is shown in Figure 4-3. The LASSO-based Selection Phase (Phase I), applies the LASSO regression with the determined regularisation parameter λ to extract a selected variable set; as shown in Eq. (4.2).

LASSO regression is used to obtain the TCF and ECF correction coefficients - as described in Eq. (4.3) and Eq. (4.4), respectively.

$$\text{y\_TCF}_i = \alpha_0 + \sum_{i=1}^{13} \alpha_i \times t_i \times fw_i \tag{4.3}$$

$$\text{y\_ECF}_i = \beta_0 + \sum_{i=1}^{8} \beta_i \times e_i \times ew_i \tag{4.4}$$

where, y_TCF_i be $\left(\frac{\text{Real\_P20}}{(\text{UAW}+\text{UUCW})\times \text{ECF}} - 0.6\right) \times \frac{1}{0.01}$, y_ECF_i be $\left(\frac{\text{Real\_P20}}{(\text{UAW}+\text{UUCW})\times \text{TCF}} - 1.4\right) \times \frac{-1}{0.03}$, and $\alpha_0, \alpha_i\ \beta_0, \beta_i$ are the regression coefficient parameters obtained from the LASSO regression; Real_P20 is the real size of software projects from historical datasets. The LASSO-based selected variables in TCF and ECF are designated as LaTF and LaEF respectively.

Then, Least Squares Regression (LSR) is used to obtain the coefficients for LaTF and LaEF in Eq. (4.5) and Eq. (4.6), respectively. LaTF and LaEF values represent the final technical and environmental complexity factors - (correction factors), in the OCF method.

$$\text{y\_LaTF}_i = \alpha_0 + \sum_{i=1}^{n} \alpha_i \times \text{LaT}_i \times \text{WLt}_i \tag{4.5}$$

$$\text{y\_LaEF}_i = \beta_0 + \sum_{i=1}^{m} \beta_i \times \text{LaE}_i \times \text{WLe}_i \tag{4.6}$$

where, let y_LaTF_i and y_LaEF_i be the TCF and ECF from the historical dataset; n is the number of LaTF; m is the number of LaEF; and $\alpha_0, \alpha_i\ \beta_0, \beta_i$ are regression coefficient parameters obtained from LSR.

LaTF and LaEF are obtained according to Eq. (4.7) and Eq. (4.8).

$$\text{LaTF} = \alpha_0 + \sum_{i=1}^{n} \alpha_i \times \text{LaT}_i \times \text{WLt}_i \tag{4.7}$$

$$\text{LaEF} = \beta_0 + \sum_{i=1}^{m} \beta_i \times \text{LaE}_i \times \text{WLe}_i \tag{4.8}$$

The model fitting phase - (Phase II) is determined. The effort estimation final result of the proposed OCF method is described by Eq. (4.9). This is calculated as the aggregate of four UAW metrics - (viz Eq. (2.1), UUCW (viz Eq. (2.2), LaTF (viz Eq. (4.7), and LaEF (viz Eq. (4.8).

$$UCP_{OCF} = (UAW + UUCW) \times LaTF \times LaEF \quad (4.9)$$

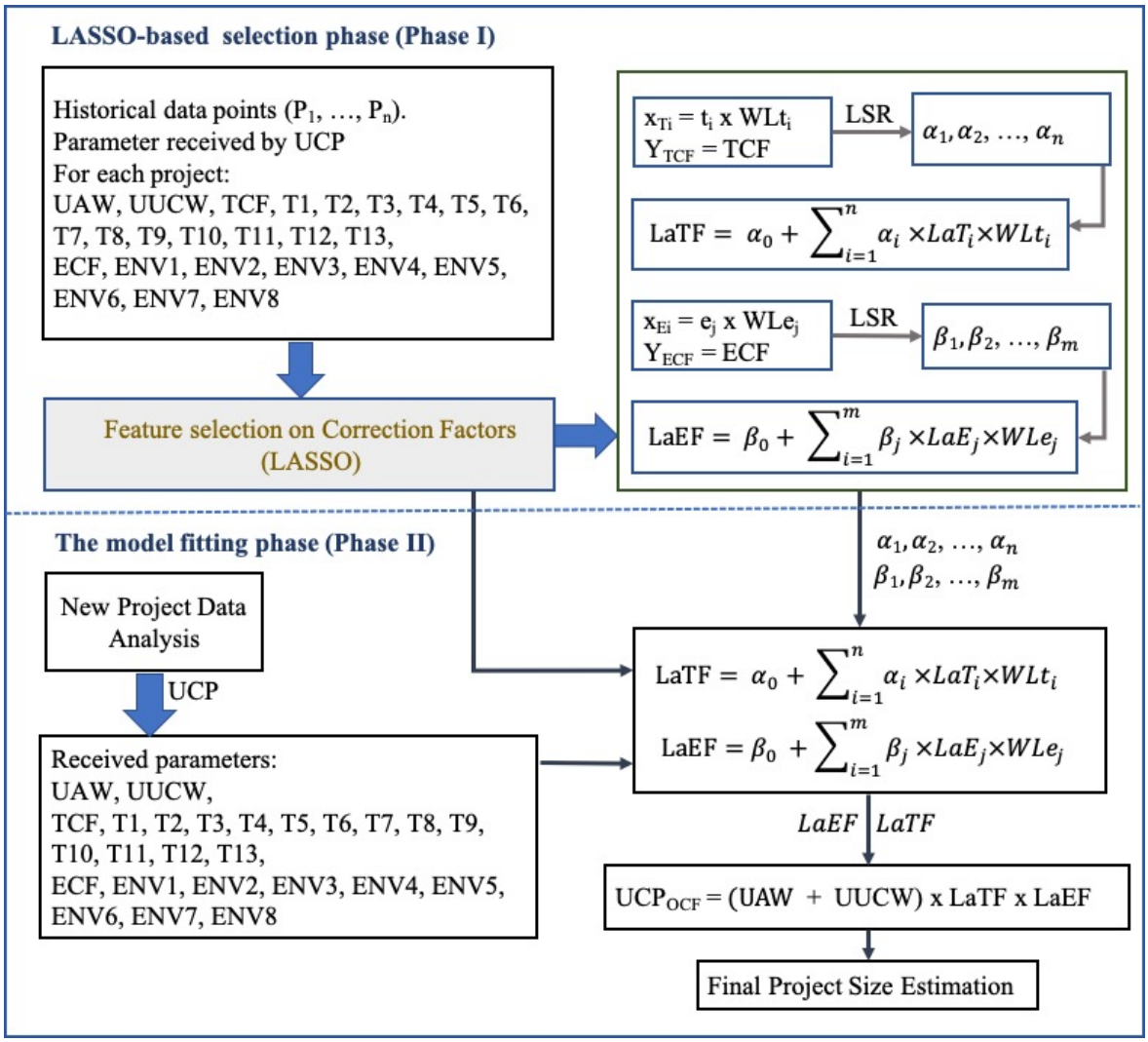


Figure 4-3. The proposed Optimizing Correction Factors method

4.2 The proposed approach based on Optimization Correction Factors and Multiple Linear Regression

The OCF approach can help project managers reduce the risks associated with evaluating correction factors. The results show that the method improves the average SSE by more than 53.6% compared to the UCP method. The detailed results are presented in Chapter 6.1. We further develop the OCF method and propose an extension of OCF (ExOCF) that applies MLR models to the OCF elements to reduce the estimation error and the influence of the unsystematic noise of the OCF technique.

4.2.1 Multiple regression models

Multiple regression models relate to estimating regression effort applications where there is more than one independent variable [103]. The purpose is to obtain the best-fit line that minimizes the regression model's sum of squared residual [19]. The form of the regression model is presented as a linear equation between a dependent variable and a set of p independent variables $X_1, X_2, \dots, X_p$ as follows:

$$\begin{cases} y_1 = \alpha_0 + \alpha_1 X_{11} + \alpha_2 X_{12} + \cdots + \alpha_p X_{1p} + \varepsilon_1 \\ y_2 = \alpha_0 + \alpha_1 X_{21} + \alpha_2 X_{22} + \cdots + \alpha_p X_{2p} + \varepsilon_2 \\ \vdots \\ y_n = \alpha_0 + \alpha_1 X_{n1} + \alpha_2 X_{n2} + \cdots + \alpha_p X_{mp} + \varepsilon_n \end{cases} \quad (4.10)$$

i.e.

$$y_i = \alpha_0 + \alpha_1 X_{i1} + \alpha_2 X_{i2} + \cdots + \alpha_p X_{ip} + \varepsilon_i, i = \overline{1 \dots m} \quad (4.11)$$

where y_i is the dependent variable, $X_{i1}, \dots, X_{ip}$ are the independent variables, α_0 is the intercept parameter, and $\alpha_1, \dots, \alpha_p$ are the regression coefficients. These variables are unknown constants that must be estimated from the dataset, and ε_i are the error residuals.

Eq. (4.10) can be rewritten as follows:

$$y = \alpha X + \varepsilon \quad (4.12)$$

where vector y and vector ε are column vectors of length m, vector α is a column vector of length $p + 1$, and matrix X is an m by $p + 1$ matrix. Using LSR, vector α is calculated as follows:

$$\alpha = (X^T X)^{-1} X^T y \quad (4.13)$$

4.2.2 Extension of Optimizing Correction Factors

The proposed method provides a straightforward approach based on historical project data to build regression models and reduce errors in the integration process or recursion. A detailed illustration of the ExOCF method is shown in Figure 4-4.

The proposed model is built using MLR as follows:

$$\begin{aligned} UCP_{ExOCF} = {} & \gamma_1 (UAW \times LaTF \times LaEF) \\ & + \gamma_2 (UUCW \times LaTF \times LaEF) \end{aligned} \quad (4.14)$$

where γ_1, γ_2 are obtained according to two steps. First, the historical data points $(P_1, \dots, P_2)$ are collected. The UAW, UUAW, LaTF, and LaEF elements for each project are identified. The result of this step is the collection of values $(x_{i1}, x_{i2}, y_i), i = \overline{1 \dots n}$, where y_i is the actual size (Real_P20 values) of the software project from a historical dataset.

$$x_{i1} = (UAW_i \times LaTF_i \times ECF_i) \quad (4.15)$$

$$x_{i2} = (UUCW_i \times LaEF_i \times ECF_i) \quad (4.16)$$

The LSR model is then used for obtaining the regression coefficients γ_1, γ_2 as followings.

$$\begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} \gamma_1 \\ \gamma_2 \end{pmatrix} \times \begin{pmatrix} X_{11} & X_{12} \\ \vdots & \vdots \\ X_{n1} & X_{n2} \end{pmatrix} \quad (4.17)$$

$$\begin{pmatrix} \gamma_1 \\ \gamma_2 \end{pmatrix} = (X^T X)^{-1} X^T y \quad (4.18)$$

Because y_i is a real value from a historical dataset, the regression coefficient values of γ_1, γ_2 can vary from each dataset. This means that when a historical dataset changes, this phase needs to be performed again to obtain new regression coefficient values. The second step of this phase will calculate the UAW, UUCW, LaTF, and LaEF of the current project, and Eq. (4.14) is applied with values γ_1, γ_2 to estimate the UCP_{ExOCF}.

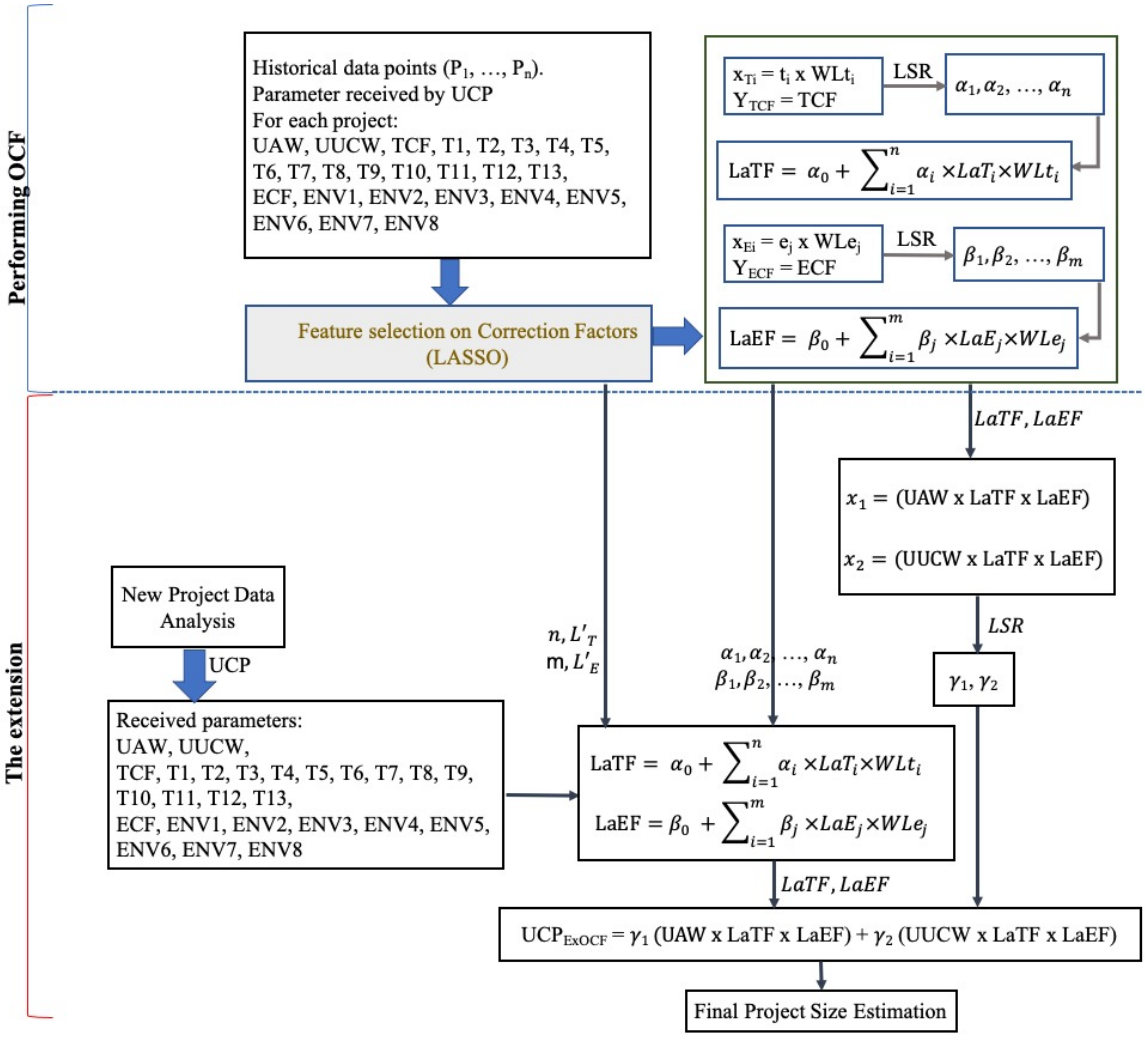


Figure 4-4. Detailed illustration of the proposed ExOCF method

4.3 The proposed Stacking ensemble model based on Optimizing Correction Factors

Based on the literature review in Chapter 3.2, we believe the ensemble approach can provide an unbiased estimate of the effort required for a new software project. The ensemble approach combines at least two different single models through a unique aggregation mechanism and generates the final solution through weighted voting on their solutions [122]. As a result, this section aims to investigate the effect of the ensemble approach in predicting the software size early in the project development using the OCF method. The SOCF model is proposed that incorporating seven statistical and ML techniques MLR, KNN, SVR, MLP, RF, GB, and DT. These techniques are presented according to their different architecture in Chapter 2.2.

4.3.1 Staking generalization approach

An ensemble of regressors is the incorporation of regressors whose individual decisions are combined to produce a system that theoretically outperforms all of its members [123]. In ensemble systems, the goal is to form a set of accurate and distinct regressors and combine their results such that the combination outperforms all individual regressors [124], [125]. Therefore, regressor ensembles are constructed in two stages: generation and combination. The individual members of the ensemble, called base regressors, are formed in the generation stage. In the combination stage, the results of the ensemble members are combined to produce a single result.

Stacked generalization (stacking) [126] uses the concept of meta-regressors or meta-learners through a learning procedure similar to cross-validation to combine the individual predictions of ensemble members. The stacking procedure is shown in general form in Figure 4-5. The stacking mechanism generates temporal estimates at level 0, called the base regressor. At level 0, the generalized biases are collectively predicted. These estimates are sent to a meta-learner at level 1.

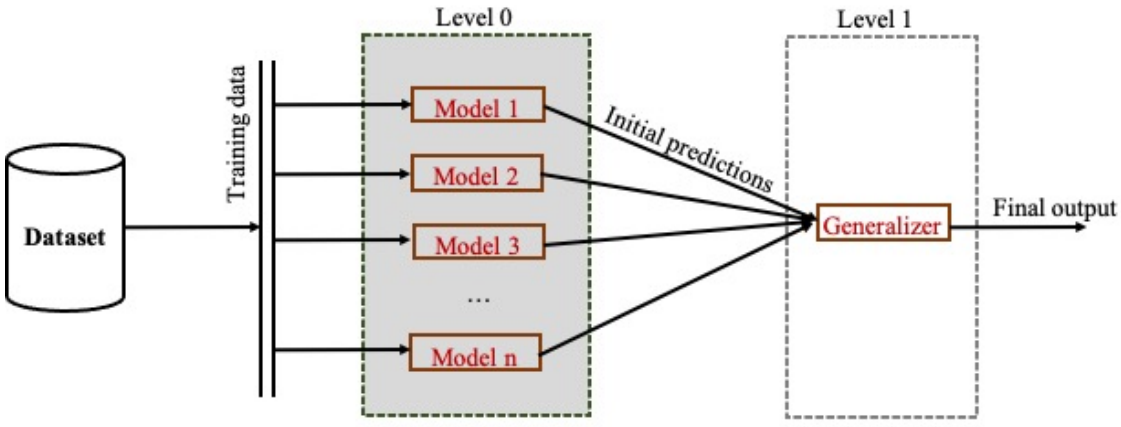


Figure 4-5. The illustration of the stacking generalization approach

4.3.2 Stacked model based on Optimizing Correction Factors

The detailed SOCF architecture is shown in Figure 4-6, which consists of cleaning the data, dividing it into training and test sets, and applying the stacking model to estimate the OCF-based size.

The following methodology was used:

1. LASSO regression is used to determine the best correction factors. Details are presented in Chapter 4.1.1.
2. The input and output vectors are determined.
3. The data is split into a training set $S^{(-j)}$ and a test set S_j. $S^{(-j)}$ is used to create the level 0 models (regressors) via seven ML techniques, SVM, KNN, DT, MLP, MLR, GB, and RF.
4. The configuration parameters for the seven regression models (level 0 models) SVM, KNN, DT, MLP, MLR, GB, and RF are tuned on the validation set (30% of the training set) to produce their optimal settings (see Section 2.2).
5. Create an ensemble model with the stacking approach. The estimator's predictions are stacked and fed into a final estimator, which computes the final estimation. More precisely, each of the level 0 models in the first stage undergo five-fold cross-validation in $S^{(-j)}$ to output its prediction and generate a prediction for S_j by taking the average of the seven estimation results generated by the five CV models in the training phase. Then these level 0 models create a vector of predictions to input into the level 1 model (in the second stage). RF was selected as the meta-regressor to train a new model for the final project size estimation.

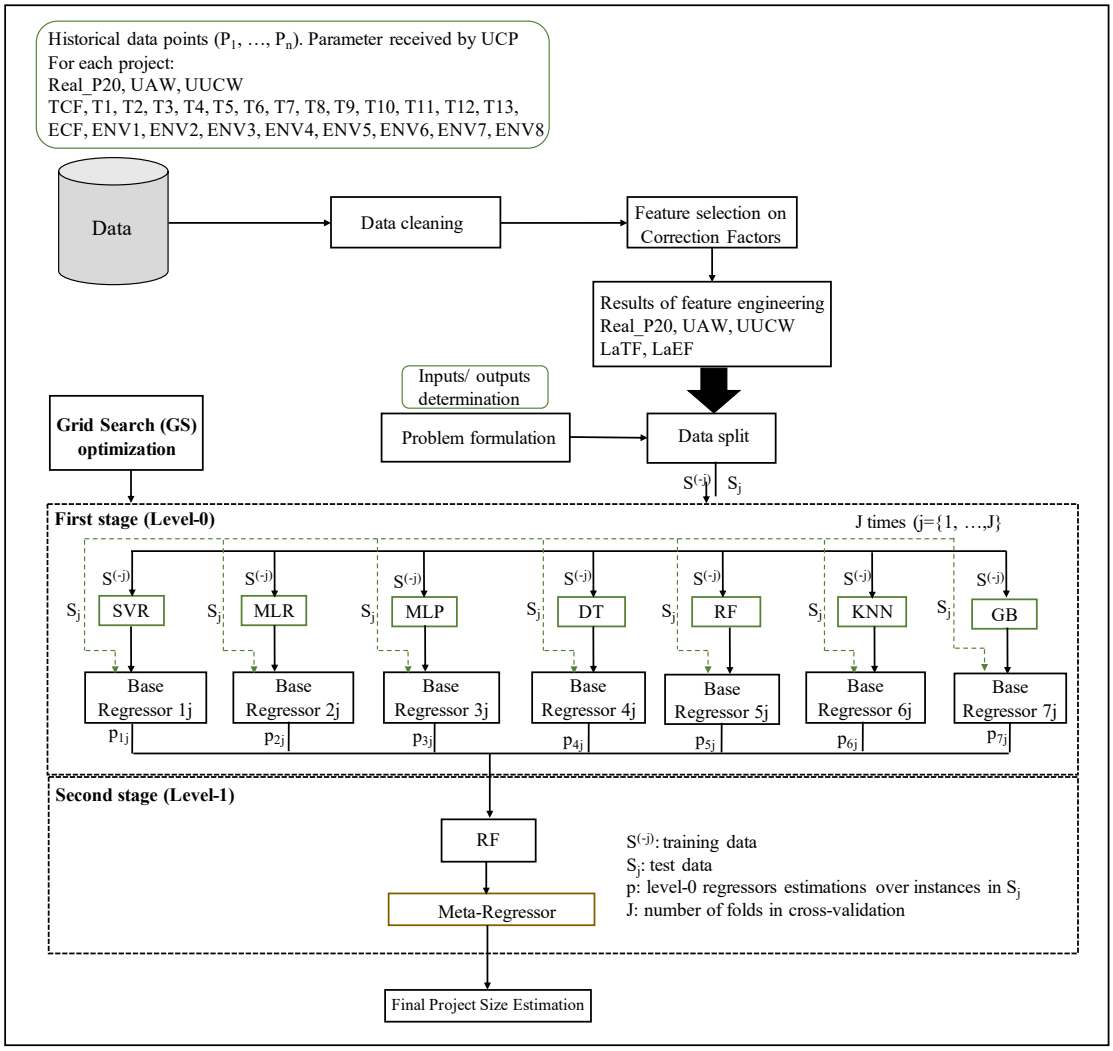


Figure 4-6. The architecture of the proposed SOCF model

4.4 The proposed software productivity model based on ensemble approach

4.4.1 Software productivity evaluation in early SDEE

Since its inception, one of the core goals of software engineering research has been to analyze and improve productivity. However, software productivity is still an issue in SDEE today, with most studies focusing on the relationship between effort and other variables. An early overview of the state of the art in software development productivity was given [127]. Since then, many studies have contributed to the knowledge of productivity factors, such as measuring productivity using multiple measures of size [46] or previous projects using case-based reasoning and regression toward the mean [42]. Multinomial logistic regression predicts productivity using independent software variables [55].

The prediction of software productivity in UCP estimation has not been comprehensively studied. Some researchers proposed to estimate effort based on

UCP using specific software productivity estimates. For example, Karner's method offered an effort by multiplying the calculated UCP size by PF, equivalent to 20 person-hours/UCP [10]. Alves et al. [47] discovered a difference between the original model (20 person-hours / UCP) and their proposed model (between 6 and 13 person-hours / UCP). Urbanek et al. [128] used linear regression to derive software productivity factors. Experimental results show that the range of productivity factors is substantially narrow than the values of UCP, with a mean of 15.

Based on a study of environmental complexity, Schneider and Winters (SW) proposed three degrees of software productivity (fair, low, and very low) [129]. More specifically, the fair level with 20 person-hours/UCP, the low level with 28 person-hours/UCP, and the very low level with 36 person-hours/UCP. These levels are determined by the total number of environmental factors with values of three or fewer from E1 to E6 and greater than three from E7 to E8. When the total count is less than or equal to two, the level is fair; when the total count is between three and four (inclusive), the level is low; and when the total count is larger than four, the level is very low. The effort of this model is calculated by Eq. (4.19). It should be noted that environmental factors are not considered when calculating UCP.

$$\text{Effort} = \begin{cases} \text{Size} \times 20, \text{if total count} \leq 2 \\ \text{Size} \times 28, \text{if } 3 \leq \text{total count} \leq 4 \\ \text{Size} \times 36, \text{if total count} > 4 \end{cases} \tag{4.19}$$

Another approach also uses environmental factors, presenting four levels of productivity measures [30]. The overall productivity factor is calculated as follows:

$$\text{PF\_sum} = \sum_{i=1}^{8} \text{E}_i \times \text{We}_i \tag{4.20}$$

This value is then converted into a productivity value using a fuzzy procedure based on the rules described in Table 4-1.

Table 4-1. Conversion rules for productivity value

No.	Antecedent	Consequent
1	$\text{PF\_sum} \leq 0$	productivity=0.4
2	$0 < \text{PF\_sum} \leq 10$	productivity=0.7
3	$10 < \text{FP\_sum} \leq 20$	productivity=1
4	$\text{PF}_{\text{sum}} > 20$	productivity=1.3

4.4.2 Effective productivity factor calculations

Our primary goal is to research and confirm the role of software productivity in generating early effort estimates from UCP. To address the fourth problem in Section 1.2, we proposed effective productivity factor calculations in conjunction with UCP as predictors for effort. The approach employs an ensemble construction mechanism from ML techniques (OCF(PF_{CFE})) such as Support Vector Regression (SVR), Multiple Linear Regression (MLR), and Decision Tree (DT). The voting ensemble is used as an ensemble model ML.

Figure 4-7 shows the proposed software productivity mode. The methodology was used: (1) Correction factors from OCF are used as input. (2) Built the voting regressor algorithm [130] consisting of three base estimators, such as Support Vector Regression (sklearn.svm.SVR), Multiple Linear Regression (sklearn.linear_model.LinearRegression), and Decision Tree Regression (sklearn.tree.DecisionTreeRegressor).

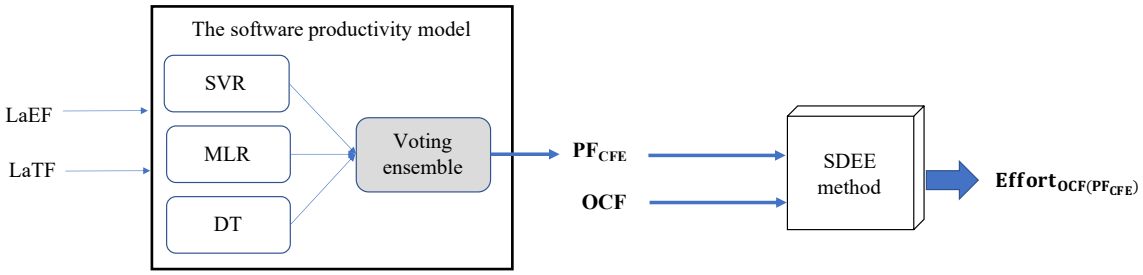


Figure 4-7. The proposed software productivity model

Estimated effort is obtained by multiplying OCF by PF_{CFE}, as follows Eq. (4.21).

$$\text{Effort}_{\text{OCF}(\text{PF}_{\text{CFE}})} = \text{OCF} \times \text{PF}_{\text{CFE}} \quad (4.21)$$

5 RESEARCH METHODOLOGY

This chapter describes the research methodology, which consists of a description of the datasets for the experiment and the empirical procedure for evaluating the proposed methods.

5.1 Dataset description

The experimental methods are evaluated using a dataset that the authors collected and used [62]. The dataset is based on three data donations (D1, D2, and D3). The projects from each data donor differ in size (measured by the UCP). All data donators work in different government, health, and business sectors. The projects were installed in Java and C# programming languages. After analyzing the dataset, we noticed that some projects had Real_P20 varied extensively. Figure 5-1 presents the boxplot of Real_P20 in each dataset. Real_P20 is a real effort in person-hours, divided by productivity (PF - person-hours per 1 UCP).

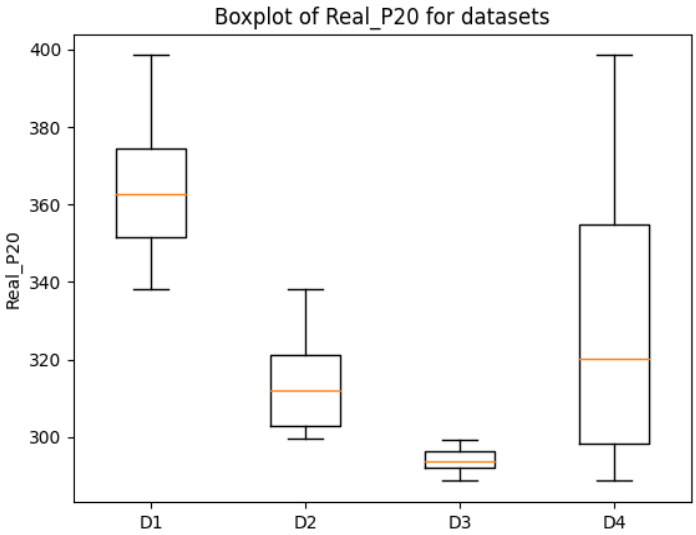


Figure 5-1. Boxplot of Real_P20 in each dataset

We discovered a significant difference in Real P20 between data donors. Depending on the data donor, the distribution of Real P20 is significantly different. In particular, data donor D1 had the largest projects, while data donor D3 had the lowest values. The dataset is heterogeneous due to the significant differences in real P20. Therefore, using the same model for all projects was ineffective. We sorted the projects by data donor to make the datasets more homogeneous. Data providers provided the datasets (D1, D2, and D3). The projects in each dataset can be interpreted as local data for each company. In addition, we evaluated the impact of combining projects with different data providers, and a fourth dataset (D4) was created to merge all three datasets.

Statistical characteristics of the Real_P20 of the four datasets are described in Table 5-1 and Figure 5-2. Median person-hours represent the workforce value of the project development period, which was applied from the project's start date to the acceptance date. The median Real_P20 shows the same value divided by PF=20. This transformation was made because data donors did not provide estimations using the UCP. The minimum Real_P20 and maximum Real_P20 describe the smallest and largest project sizes, respectively. The Real_P20 range describes the difference between the minimum Real_P20 and maximum Real_P20. The "n" indicates the number of projects in the dataset.

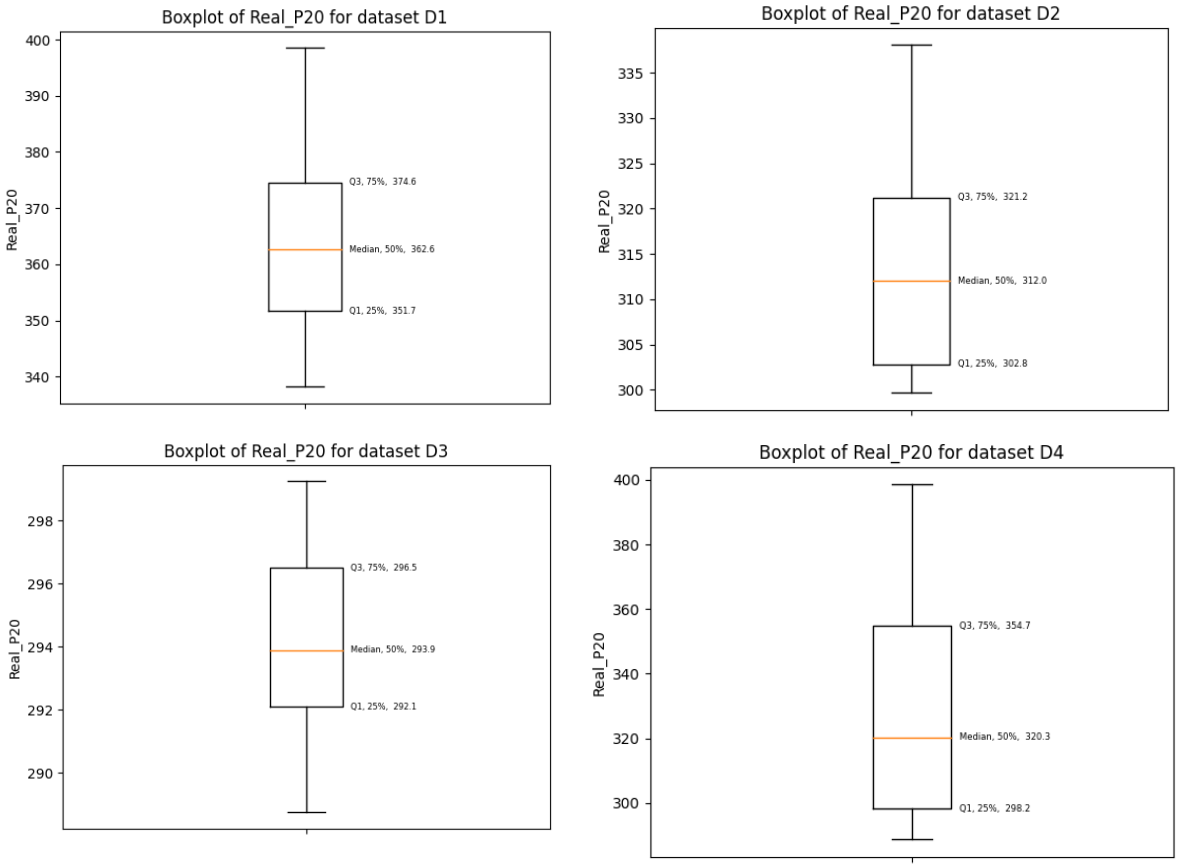


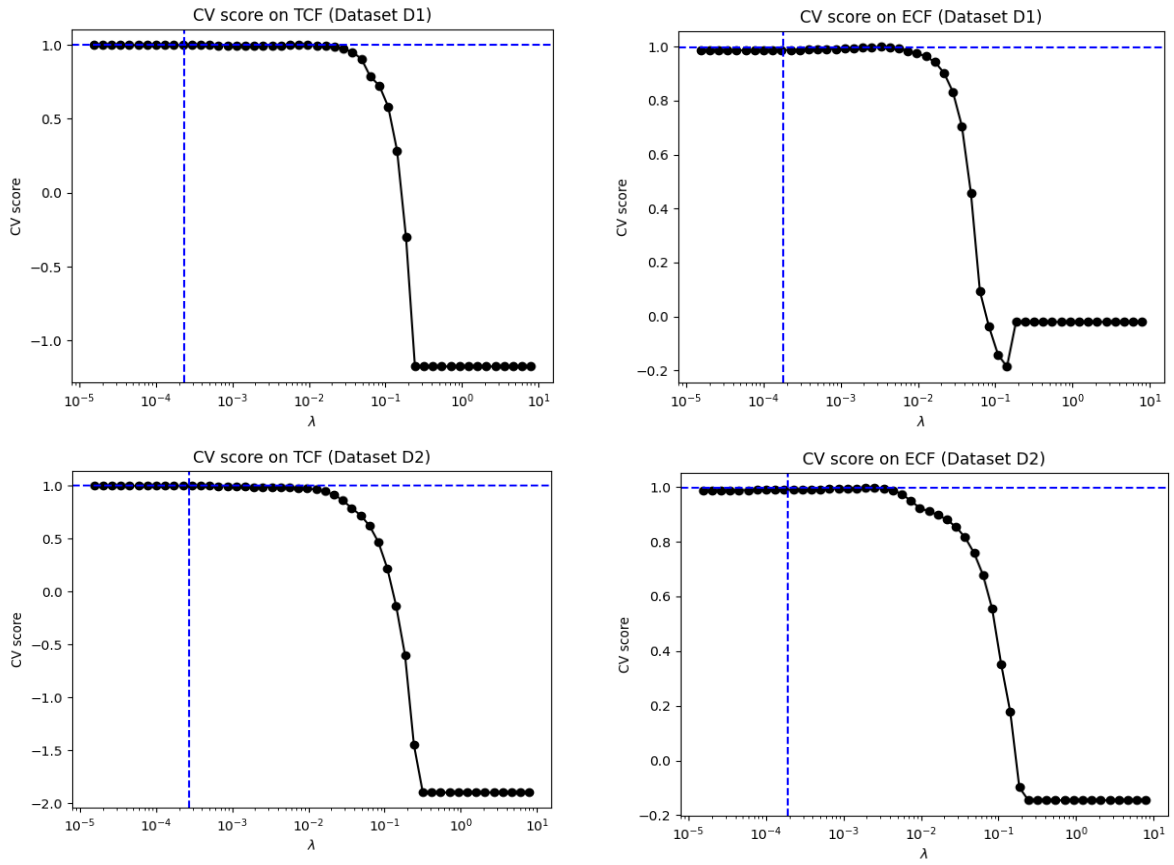
Figure 5-2. Statistical characteristics of the Real_P20 for each dataset

Table 5-1. Dataset statistical characteristics

	Median person-hours	Median Real_P20	Range Real_P20	Standard deviation	Minimum Real_P20	Maximum Real_P20	n
D1	7252.0	362.600	60.300	18.820	338.200	398.500	27
D2	6240.0	312.000	38.400	12.156	299.650	338.050	23
D3	5878.0	293.900	10.500	3.287	288.750	299.250	20
D4	6406.0	320.300	109.750	33.212	288.750	398.500	70

5.2 Correction factors determination

Based on the analysis described in section 4.1.2, we solved this problem using the LASSO to determine correction factors in the regression analysis. Figure 5-3 shows a sequence of different R-squared values in proportion to different values of λ. The selected λ value is determined using the Leave One Out Cross-Validation (LOOCV) technique [131], [132] at which the R-squared reaches its highest value. Figure 5-4 shows the selected technical and environmental factors corresponding to the determined λ values.



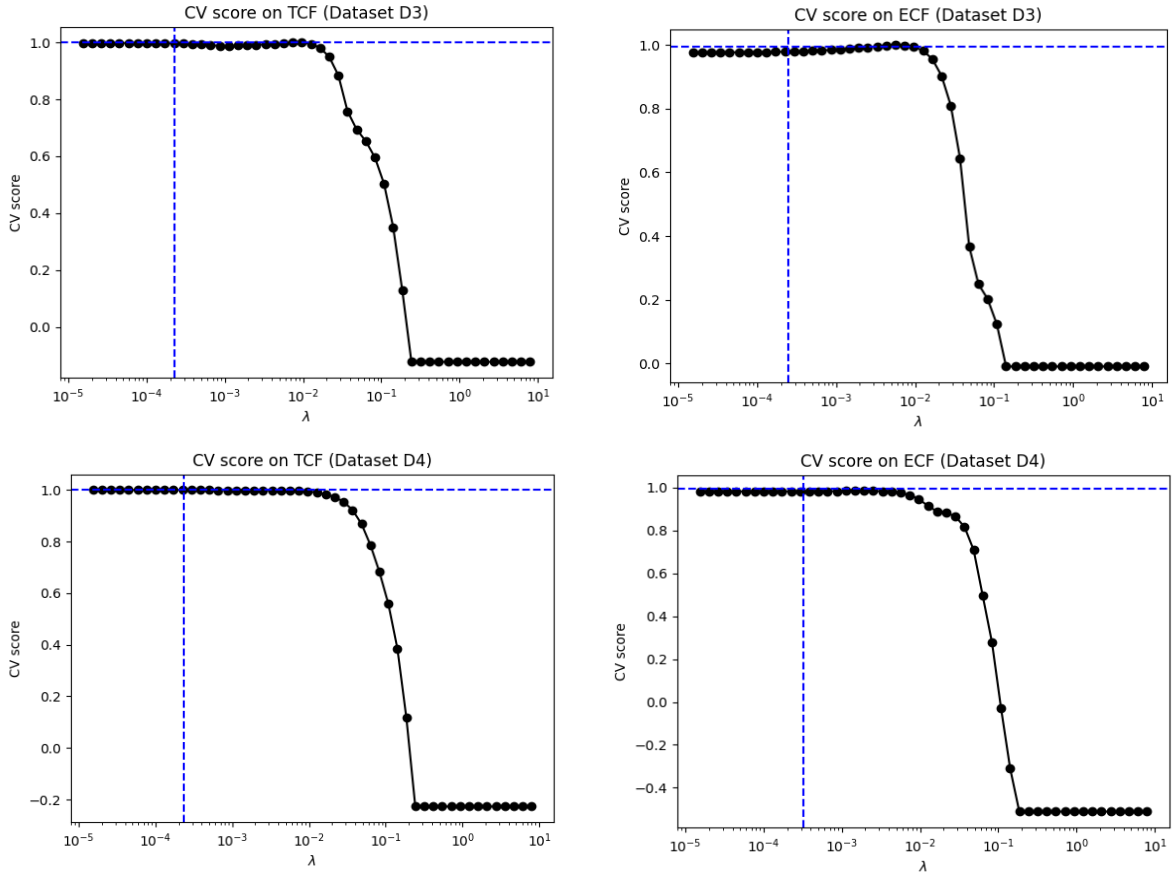


Figure 5-3. CV score on TCF and ECF in each dataset

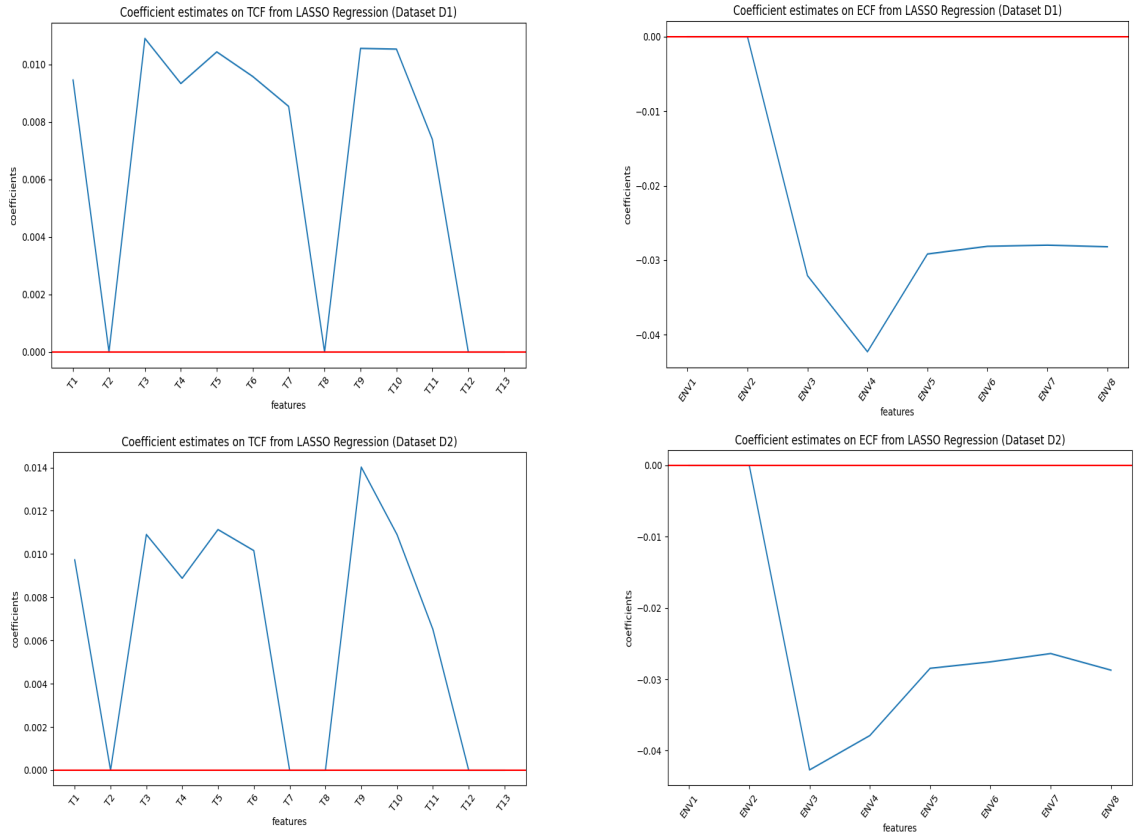


Figure 5-4. Coefficient estimations on TCF and ECF from LASSO regression in each dataset

The details of the technical and environmental factors selected in each dataset with the determined, as well as their coefficient estimates, are shown in Table 5-2 and Table 5-3. Specifically, there are nine remaining technical correction factors in the D1 dataset at T1, T3, T4, T5, T6, T7, T9, T10, and T11, and there are six remaining environmental factors, ENV3 to ENV8. In the D2 dataset, the eight selected technical factors are T1, T3, T4, T5, T6, T9, T10, and T11, and the selected environmental factors are ENV3 to ENV8. In the D3 dataset, the seven selected technical factors are T1, T3, T4, T5, T7, T9, and T10, and the selected environmental factors at 0.000247 are ENV3 to ENV8. In the D4 dataset, the nine selected technical factors are T1, T3, T4, T5, T6, T7, T9, T10, and T11, and the environmental factors are ENV3-ENV8.

Table 5-2. The estimated TCF coefficients in the LASSO regression

	D1	D2	D3	D4
	0.000231	0.000268	0.000227	0.000236
intercept	0.690619	0.693400	0.720820	0.695850
T1	0.009451	0.009725	0.009547	0.009505
T2	-	-	-	-
T3	0.010897	0.010902	0.010311	0.010456
T4	0.009330	0.008877	0.009888	0.009556
T5	0.010430	0.011130	0.015199	0.010622
T6	0.009576	0.010157	-	0.009202
T7	0.008536	-	0.007298	0.008989
T8	-	-	-	-
T9	0.010551	0.014018	0.013144	0.010334
T10	0.010526	0.010893	0.009730	0.010902
T11	0.007387	0.006516	-	0.005998
T12	-	-	-	-
T13	-	-	-	-

Table 5-3. The estimated ECF coefficients in the LASSO regression

	D1	D2	D3	D4
	0.000177	0.000192	0.000247	0.000327
intercept	1.373478	1.376197	1.404496	1.387716
ENV1	-	-	-	-
ENV2	-	-	-	-
ENV3	-0.032072	-0.042706	-0.032954	-0.033555
ENV4	-0.042291	-0.037886	-0.025558	-0.033001
ENV5	-0.029170	-0.028453	-0.029931	-0.029393
ENV6	-0.028133	-0.027549	-0.030139	-0.029072
ENV7	-0.027981	-0.026382	-0.029221	-0.028660
ENV8	-0.028193	-0.028713	-0.031169	-0.029333

5.3 Experiment setup

A series of experimental setups are presented to evaluate the effectiveness of the proposed methods. In step 1, the proposed methods in this research direction are established for the following experiments:

- OCF (proposed in Chapter 4.1)
- ExOCF (proposed in Chapter 4.2)
- SOCF (proposed in Chapter 4.3)
- OCF(PF_{CFE}) (proposed in Chapter 4.4)

We experimented with five different runs (5-fold cross-validation) to evaluate the estimation accuracy. The comparison of each method's effort estimation accuracy is then based on the average results of these five runs.

In step 2, the results were then evaluated using evaluation criteria, SSE, PRED (0.25), MAE, MBRE, MIBRE, MdMRE, and RMSE, as presented in Chapter 2.3. A statistical pairwise t-test comparison (at a 5% significance level) was also used to validate the accuracy of the proposed methods. These pairwise statistical comparisons include the average (μ) of the SSE, MAE, and RMSE results from the five-fold cross-validations of the four experimental datasets.

5.3.1 Experiment 1 (EX1)

EX1 is performed to evaluate the proposed OCF method with other related methods, such as the baseline UCP [10] and OTF - a variant of the UCP model that omits the technical factors [133]. These methods are summarized in Table 5-4.

Table 5-4. Methods implemented for EX1

No.	SDEE methods	Summary	Notation
1	Use Case Points	- Size is measured by UCP variables (UAW, UUCW, TCF, and ECF).	UCP
2	UCP (omitting technical factors)	- Size is measured from UCP variables (UAW, UUAW, and ECF) except for the technical factors.	OTF

3	Optimization Correction Factors (proposed in Chapter 4.1)	- Correction factors are determined in regression analysis by the LASSO regression model. - Size is measured in UCP variables (UAW and UUCW) and correction factors (LaTF and LaEF).	OCF

The statistical hypothesis was tested to determine whether the proposed OCF approach provides a better estimate.

- H_0: $\mu_{\text{the proposed OCF method}} = \mu_{\text{the other tested methods}}$. The estimation ability of the proposed OCF method is not significantly different from the estimation abilities of the other tested methods.
- H_1: $\mu_{\text{the proposed OCF method}} > \mu_{\text{the other tested methods}}$. The estimation ability of the proposed OCF method is significantly different from the estimation abilities of the other tested methods.

5.3.2 Experiment 2 (EX2)

EX2 is performed to evaluate the proposed ExOCF method with the related software size estimation models from the literature. The selected models are the baseline UCP model [10], the OCF model, and the AOM model [23]. We also developed two models that establish a linear relationship between software and UCP factors (UAW, UUC, TCF, and EF). These models are SVR, and DT. The description of the construction of these models is mentioned in Chapter 2.2. However, in this experiment, we do not focus on finding the optimal configuration parameters but use the default configuration parameters of each model. All methods are summarized in Table 5-5.

Table 5-5. Methods implemented for EX2

No.	SDEE method	Summary	Notation
1	Use Case Points	- Size is measured by UCP variables (UAW, UUCW, TCF, and ECF).	UCP

2	Optimization Correction Factors (proposed in Chapter 4.1)	- Correction factors are determined in regression analysis by the LASSO regression model. - Size is measured in UCP size variables (UAW and UUCW) and correction factors (LaTF and LaEF).	OCF
3	Algorithmic Optimization Method	- Size is measured based on linear regression on UCP variables (UAW, UUC, TCF, and EF).	AOM
4	Use Case Points using SVR	- SVR is used to estimate the software size based on UCP variables (UAW, UUCW, TCF, and ECF).	UCP&SVR
5	Use Case Points using DT	- DT is used to estimate the software size based on UCP variables (UAW, UUCW, TCF, and ECF).	UCP&DT
6	Extension of Optimization Correction Factors (proposed in Chapter 4.2)	- Correction factors are determined in regression analysis by the LASSO regression model. - Size is based on linear regression on OCF variables (UAW, UUCW, LaTF, and LaEF).	ExOCF

The statistical hypothesis was tested to determine whether the proposed ExOCF approach provides a better estimate.

- H_0: $\mu_{\text{the proposed ExOCF method}} = \mu_{\text{the other tested methods}}$. The estimation ability of the proposed ExOCF method is not significantly different from the estimation abilities of the other tested methods.
- H_1: $\mu_{\text{the proposed ExOCF method}} > \mu_{\text{the other tested methods}}$. The estimation ability of the proposed ExOCF method is significantly different from the estimation abilities of the other tested methods.

5.3.3 Experiment 3 (EX3)

EX3 is conducted to compare the proposed SOCF method with the related SDEE methods, such as UCP-based single methods (described in Table 5-6),

OCF-based single methods (described in Table 5-7), and ensemble methods (described in Table 5-8). In addition, we experimented with baseline SDEE methods (UCP and OCF).

Table 5-6. UCP-based single methods implemented for EX3

No.	ML technique	Summary	Notation
1	SVR	- SVR is used to estimate the software size based on UCP variables (UAW, UUCW, TCF, and ECF).	UCP&SVR
2	KNN	- KNN is used to estimate the software size based on UCP variables (UAW, UUCW, TCF, and ECF).	UCP&KNN
3	DT	- DT is used to estimate the software size based on UCP variables (UAW, UUCW, TCF, and ECF).	UCP&DT
4	GRNN	- GRNN is used to estimate the software size based on UCP variables (UAW, UUCW, TCF, and ECF).	UCP&GRNN
5	MLP	- MLP is used to estimate the software size based on UCP variables (UAW, UUCW, TCF, and ECF).	UCP&MLP
6	RF	- RF is used to estimate the software size based on UCP variables (UAW, UUCW, TCF, and ECF).	UCP&RF

Table 5-7. OCF-based single methods implemented for EX3

No.	ML technique	Summary	Notation
1	SVR	- SVR is used to estimate the software size based on OCF variables (UAW, UUCW, LaTF, and LaEF).	OCF&SVR

2	MLP	- MLP is used to estimate the software size based on OCF variables (UAW, UUCW, LaTF, and LaEF).	OCF&MLP
3	GB	- GB is used to estimate the software size based on OCF variables (UAW, UUCW, LaTF, and LaEF).	OCF&GB
4	MLR	- MLR is used to estimate the software size based on OCF variables (UAW, UUCW, LaTF, and LaEF).	OCF&MLR
5	KNN	- KNN is used to estimate the software size based on OCF variables (UAW, UUCW, LaTF, and LaEF).	OCF&KNN
6	DT	- DT is used to estimate the software size based on OCF variables (UAW, UUCW, LaTF, and LaEF).	OCF&DT
7	RF	- RF is used to estimate the software size based on OCF variables (UAW, UUCW, LaTF, and LaEF).	OCF&RF

Table 5-8. Ensemble methods implemented for EX3

No.	ML technique	Summary	Notation
1	Majority voting ensemble	- Majority voting ensemble with MLR, SVR, MLP models to estimate the software size based on UCP variables (UAW, UUCW, TCF, and ECF).	VUCP
2	Stacked Generalization Ensemble	- Stacked generalization ensemble with SVM, KNN, DT, MLP, MLR, GB, RF models to estimate the software size based on OCF variables (UAW, UUCW, LaTF, and LaEF).	SOCF (proposed in Chapter 4.3)

The accuracy of a given statistical or ML technique is determined by the configuration parameters that describe the characteristics of a specific dataset. Choosing the optimal parameter values for a technique gives it a higher predictive ability. In this EX3, we apply GS [134] to optimize the configuration settings of statistical and ML techniques. Specifically, for each dataset, GS thoroughly examines the parameter set of each empirical method in a given range of values and then selects the configuration that provides optimal estimates. The parameter search ranges are derived from previous analyses [54], [55], [59]. In each case, we extended the search range to include as many alternative configurations as possible. The convergence of each optimization approach is determined by the mean square error (MSE) reaching zero or the maximum number of iterations reaching 10,000 [135]. The settings are tuned to the validation set, which comprises 30% of the training set. Table 5-9, Table 5-10, Table 5-9, and Table 5-10 list the optimal parameter values for each dataset's estimation method.

Table 5-9. The results of parameter tunings in the D1 dataset

Method	Parameters settings
OCF&MLP	$\mathrm{L} = 0.05, \mathrm{H} = 7, \mathrm{M} = 0.2, \alpha = 0.0001$
UCP&MLP	$\mathrm{L} = 0.04, \mathrm{H} = 7, \mathrm{M} = 0.5, \alpha = 0.001$
OCF&SVR	$\mathrm{C} = 10, \gamma = 0.001, \varepsilon = 0.001$
UCP&SVR	$\mathrm{C} = 10, \gamma = 1, \varepsilon = 1$
OCF&DT	$\mathrm{max\_depth} = 7, \mathrm{min\_weight\_fraction\_leaf} = 0.4,$ $\mathrm{max\_leaf\_node} = 40, \mathrm{min\_sample\_leaf} = 10$
UCP&DT	$\mathrm{max\_depth} = 5, \mathrm{min\_weight\_fraction\_leaf} = 0.3,$ $\mathrm{max\_leaf\_node} = 20, \mathrm{min\_sample\_leaf} = 6$
OCF&RF	$\mathrm{n\_estimators} = 100, \mathrm{min\_sample\_leaf} = 2,$ $\mathrm{max\_depth} = 10$
UCP&RF	$\mathrm{n\_estimators} = 150, \mathrm{min\_sample\_leaf} = 2,$ $\mathrm{max\_depth} = 20$
OCF&GB	$\mathrm{n\_estimators} = 60, \mathrm{min\_sample\_leaf} = 60,$ $\mathrm{max\_depth} = 5$
OCF&KNN	$\mathrm{neighbors} = 5$
UCP&KNN	$\mathrm{neighbors} = 10$
UCP&GRNN	$\sigma = 0.1$

Table 5-10. The results of parameter tunings in the D2 dataset

Method	Parameters settings
OCF&MLP	$\mathrm{L}=0.02, \mathrm{H}=8, \mathrm{M}=0.5, \alpha=0.001$
UCP&MLP	$\mathrm{L}=0.03, \mathrm{H}=6, \mathrm{M}=0.5, \alpha=0.0001$
OCF&SVR	$\mathrm{C}=100, \gamma=0.1, \varepsilon=0.001$
UCP&SVR	$\mathrm{C}=10, \gamma=1, \varepsilon=0.1$
OCF&DT	$\max\_depth=5, \min\_weight\_fraction\_leaf=0.5,$ $\max\_leaf\_node=30, \min\_sample\_leaf=4$
UCP&DT	$\max\_depth=3, \min\_weight\_fraction\_leaf=0.3,$ $\max\_leaf\_node=40, \min\_sample\_leaf=2$
OCF&RF	$\mathrm{n\_estimators}=200, \min\_sample\_leaf=1,$ $\max\_depth=50$
UCP&RF	$\mathrm{n\_estimators}=100, \min\_sample\_leaf=1,$ $\max\_depth=30$
OCF&GB	$\mathrm{n\_estimators}=20, \min\_sample\_leaf=40,$ $\max\_depth=6$
OCF&KNN	$\mathrm{neighbors}=8$
UCP&KNN	$\mathrm{neighbors}=9$
UCP&GRNN	$\sigma=0.3$

Table 5-11. The results of parameter tunings in the D3 dataset

Method	Parameters settings
OCF&MLP	$\mathrm{L}=0.01, \mathrm{H}=6, \mathrm{M}=0.2, \alpha=0.01$
UCP&MLP	$\mathrm{L}=0.04, \mathrm{H}=6, \mathrm{M}=0.2, \alpha=0.01$
OCF&SVR	$\mathrm{C}=50, \gamma=1, \varepsilon=1$
UCP&SVR	$\mathrm{C}=10, \gamma=0.01, \varepsilon=0.01$
OCF&DT	$\max\_depth=9, \min\_weight\_fraction\_leaf=0.3,$ $\max\_leaf\_node=10, \min\_sample\_leaf=5$
UCP&DT	$\max\_depth=5, \min\_weight\_fraction\_leaf=0.1,$ $\max\_leaf\_node=30, \min\_sample\_leaf=7$

OCF&RF	$\text{n\_estimators} = 300, \text{min\ \_sample\_leaf} = 4,$ $\text{max\ \_depth} = 80$
UCP&RF	$\text{n\_estimator} = 400, \text{min\ \_sample\_leaf} = 2,$ $\text{max\ \_depth} = 50$
OCF&GB	$\text{n\_estimators} = 30, \text{min\ \_sample\_leaf} = 30,$ $\text{max\ \_depth} = 7$
OCF&KNN	$\text{neighbors} = 6$
UCP&KNN	$\text{neighbors} = 10$
UCP&GRNN	$\sigma = 0.6$

Table 5-12. The results of parameter tunings in the D4 dataset

Method	Parameters settings
OCF&MLP	$\text{L} = 0.02, \text{H} = 6, \text{M} = 0.3, \alpha = 0.01$
UCP&MLP	$\text{L} = 0.03, \text{H} = 6, \text{M} = 0.4, \alpha = 0.001$
OCF&SVR	$\text{C} = 100, \gamma = 1, \varepsilon = 0.01$
UCP&SVR	$\text{C} = 10, \gamma = 0.01, \varepsilon = 0.01$
OCF&DT	$\text{max\ \_depth} = 3, \text{min\ \_weight\_fraction\_leaf} = 0.1,$ $\text{max\_leaf\_node} = 50, \text{min\_sample\_leaf} = 2$
UCP&DT	$\text{max\ \_depth} = 5, \text{min\ \_weight\_fraction\_leaf} = 0.5,$ $\text{max\_leaf\_node} = 30, \text{min\_sample\_leaf} = 3$
OCF&RF	$\text{n\_estimators} = 250, \text{min\ \_sample\_leaf} = 4,$ $\text{max\ \_depth} = 10$
UCP&RF	$\text{n\_estimators} = 300, \text{min\ \_sample\_leaf} = 4,$ $\text{max\ \_depth} = 20$
OCF&GB	$\text{n\_estimators} = 40, \text{min\ \_sample\_leaf} = 50,$ $\text{max\ \_depth} = 6$
OCF&KNN	$\text{neighbors} = 7$
UCP&KNN	$\text{neighbors} = 9$
UCP&GRNN	$\sigma = 0.4$

The statistical hypothesis was tested to determine whether the proposed SOCF approach provides a better estimate.

- H_0: $\mu_{\text{the proposed SOCF method}} = \mu_{\text{the other tested methods}}$. The estimation ability of the proposed SOCF method is not significantly different from the estimation abilities of the other tested methods.
- H_1: $\mu_{\text{the proposed SOCF method}} > \mu_{\text{the other tested methods}}$. The estimation ability of the proposed SOCF method is significantly different from the estimation abilities of the other tested methods.

5.3.4 Experiment 4 (EX4)

EX4 is conducted to compare the proposed OCF(PFCFE) method with the previous SDEE methods (UCP, SW [129], OCF), as summarized in Table 5-13. Table 5-14 presents the optimal configuration parameters in the experiments.

Table 5-13. Methods implemented for EX4

No.	SDEE method	Summary	Notation
1	Use Case Points	- Size is measured by UCP variables (UAW, UUCW, TCF, and ECF). - 20 person-hours to develop each UCP (PF=20). - The effort is computed by multiplying Size by the PF.	UCP
2	Schneider and Winter (SW)	- Size is measured by UCP variables (UAW, UUCW, TCF, and ECF). - PF is computed from the UCP environmental complexity factors. - The effort is computed by multiplying Size by the PF.	SW
3	Optimization Correction Factors	- Correction factors are determined in regression analysis by the LASSO regression model. - Size is measured in UCP size variables (UAW and UUCW) and correction factors (LaTF and LaEF).	OCF

		- 20 person-hours to develop each UCP (PF=20). - The effort is computed by multiplying Size by the PF.	
4	Software Productivity Model based on Ensemble Construction Mechanism (proposed in Chapter 4.4)	- Size is measured in UCP size variables (UAW and UUCW) and correction factors (LaTF and LaEF). - A proposed PF_{CFE} model is constructed based on correction factors through an ensemble construction mechanism of three ML models (SVR, MLR, and DT). Details of the optimal configuration parameter sets after tuning are shown in Table 5-14. - The effort is computed by multiplying Size by the $PF_{CFE.}$	$OCF(PF_{CFE})$

Table 5-14. The optimal values of method parameters in EX4

Method	Parameters setting
	D1 dataset
PF_{SVR}	$C = 0.1, \gamma = 0.0001, \varepsilon = 0.1$
PF_{DT}	$\max\_depth = 7, \min\_weight\_fraction\_leaf = 0.4$, $max\_leaf\_node = 70, min\_sample\_leaf = 7$
	D2 dataset
PF_{SVR}	$C = 0.01, \gamma = 0.0001, \varepsilon = 0.01$
PF_{DT}	$\max\_depth = 9, \min\_weight\_fraction\_leaf = 0.1$, $max\_leaf\_node = 80, min\_sample\_leaf = 4$
	D3 dataset
PF_{SVR}	$C = 1, \gamma = 0.01, \varepsilon = 0.001$
PF_{DT}	$\max\_depth = 4, \min\_weight\_fraction\_leaf = 0.3$, $max\_leaf\_node = 30, min\_sample\_leaf = 2$

	D4 dataset
PF_{SVR}	$C = 1, \gamma = 0.0001, \varepsilon = 0.1$
PF_{DT}	$\max\_depth = 3, \min\_weight\_fraction\_leaf = 0.2$, $max\_leaf\_node = 50, min\_sample\_leaf = 7$

The statistical hypothesis was tested to determine whether the proposed OCF(PF_{CFE}) approach provides a better estimate.

- H_0: $\mu_{\text{the proposed OCF(PFCFE) method}} = \mu_{\text{the other tested methods}}$. The estimation ability of the proposed SOCF method is not significantly different from the estimation abilities of the other tested methods.
- H_1: $\mu_{\text{the proposed OCF(PFCFE) method}} > \mu_{\text{the other tested methods}}$. The estimation ability of the proposed SOCF method is significantly different from the estimation abilities of the other tested methods.

6 RESULTS AND DISCUSSION

This section presents the solutions to the four problem statements given above. The purpose of the results is to minimize the SSE, MdMRE, MAE, MBRE, MIBRE, and RMSE and maximize the PRED (0.25). Specifically, low values for the SSE, MdMRE, MAE, MBRE, MIBRE, and RMSE show good results. In contrast, high values for the PRED (0.25) show good results. Besides that, the results of SSE, MAE, MdMRE, MBRE, MIBRE, and RMSE in the four experimental datasets were used for the paired t-test statistical comparisons. After five runs on different random training- testing had split, we obtained the average p-value of the t-test.

6.1 EX1

In the EX1, we will compare the proposed OCF method as well as the UCP and OTF methods based on the four experimental datasets. Table 6-1, Table 6-2, Table 6-3, and Table 6-4 show that the proposed OCF method outperformed the UCP and OTF methods when the SSE, MAE, MBRE, MIBRE, MdMRE, and RMSE criteria were used. The OCF method also gave good results when PRED (0.25) was used. Figure 6-1 shows the average estimation results of the proposed OCF and other methods.

Based on the estimation results in Table 6-1, Table 6-2, Table 6-3, and Table 6-4, we present the percentage improvements of the proposed OCF over the UCP and OTF methods averaged on all datasets in Table 6-5.

Table 6-1. Estimation results for the proposed OCF method and other methods on the D1 dataset

Methods	SSE	PRED	MAE	MdMRE	MBRE	MIBRE	RMSE
UCP	54,838.961	0.40	95.615	0.281	0.366	0.252	104.339
OTF	29,971.565	0.68	49.904	0.206	0.241	0.187	77.056
OCF	14,215.263	0.92	44.784	0.112	0.146	0.120	52.964

Table 6-2. Estimation results for the proposed OCF method and other methods on the D2 dataset

Methods	SSE	PRED	MAE	MdMRE	MBRE	MIBRE	RMSE
UCP	34,148.372	0.55	82.544	0.254	0.384	0.252	92.152
OTF	23,153.238	0.60	64.267	0.223	0.268	0.202	75.232
OCF	17,773.295	0.65	54.393	0.165	0.227	0.167	65.596

Table 6-3. Estimation results for the proposed OCF method and other methods on the D3 dataset

Methods	SSE	PRED	MAE	MdMRE	MBRE	MIBRE	RMSE
UCP	58,932.202	0.47	120.02	0.455	0.426	0.278	138.640
OTF	49,847.551	0.47	111.72	0.422	0.389	0.260	127.000
OCF	30,148.049	0.53	87.242	0.266	0.331	0.237	95.369

Table 6-4. Estimation results for the proposed OCF method and other methods on the D4 dataset

Methods	SSE	PRED	MAE	MdMRE	MBRE	MIBRE	RMSE
UCP	59,995.454	0.33	125.85	0.419	0.441	0.289	139.698
OTF	51,670.526	0.33	113.27	0.414	0.395	0.262	129.075
OCF	34,210.337	0.53	87.308	0.261	0.318	0.225	98.899

Table 6-5. The percentage improvements of the OCF over the UCP and OTF methods averaged on all datasets

Methods	SSE	PRED	MAE	MdMRE	MBRE	MIBRE	RMSE
OCF vs.UCP	53.6%	33.6%	35.4%	42.9%	36.7%	30.2%	34.1%
OCF vs.OTF	37.7%	21.1%	19.2%	36.4%	20.8%	17.8%	23.3%

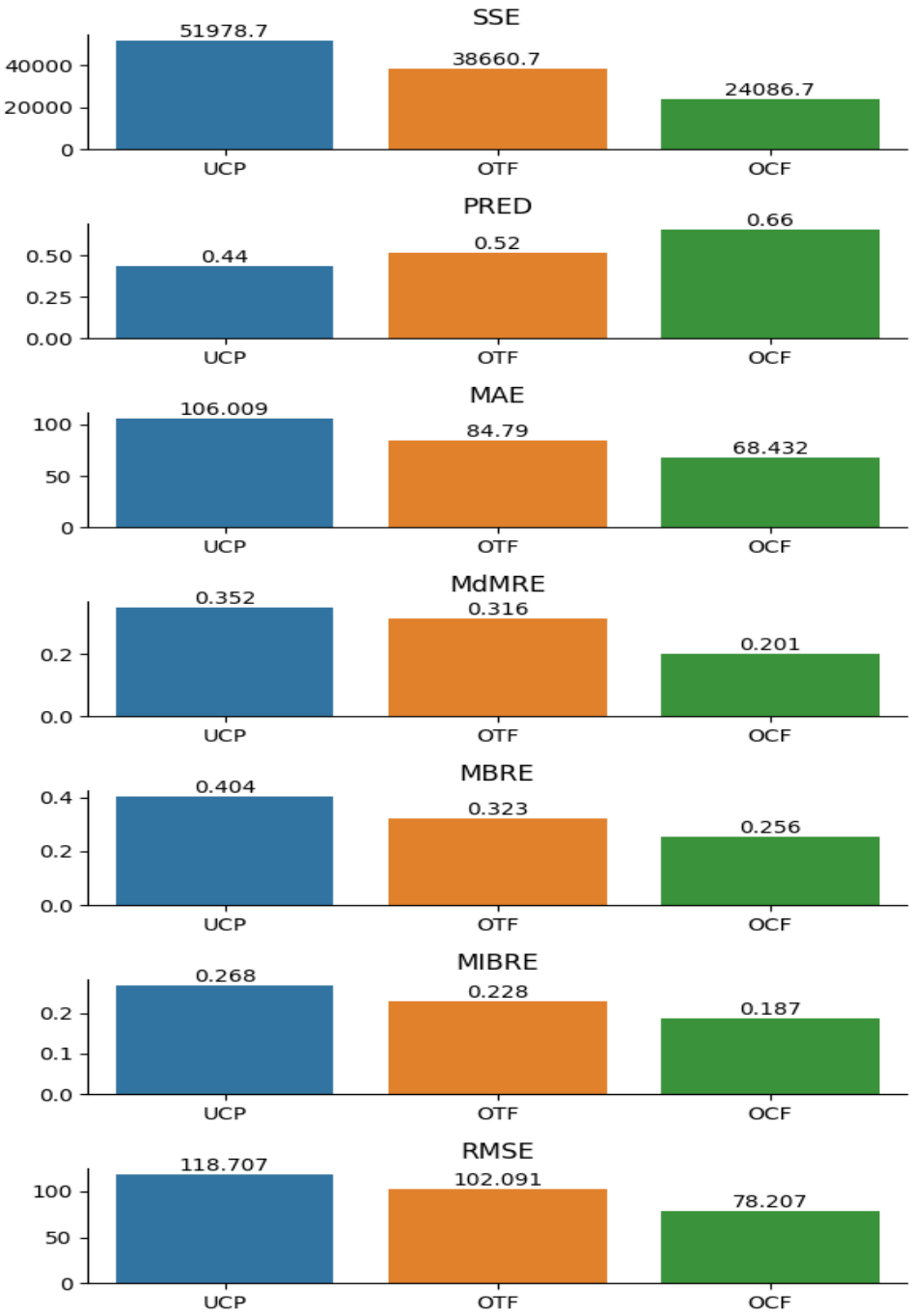


Figure 6-1. The average estimation results of the proposed OCF method and other methods on all datasets

Moreover, we use the SSE, MAE, and RMSE results for all the experimental methods for statistical comparisons, i.e., to draw the most accurate conclusions by comparing estimation methods. The t-test, a parametric statistical comparison test, is used in this study. For a less than, the two statistical methods involved in the comparison are significantly different. As shown in Table 6-6, our proposed OCF method is statistically superior to the baseline UCP method and the OTF

method. A>> B means that A is statistically superior to B. Therefore, we accept the alternative hypothesis H1.

Table 6-6. The t-test results for five different runs of the proposed OCF method in comparison with the other methods.

Pairs of methods		OCF vs. UCP	OCF vs. OTF
SSE	Avg. SSE	24,086.736 vs. 51978.747	24,086.736 vs. 38660.720
	Avg. p-value	0.00000	0.00000
	Statistical conclusion	>>	>>
MAE	Avg. MAE	68.432 vs. 106.009	68.432 vs. 84.790
	Avg. p-value	0.00000	0.00001
	Statistical conclusion	>>	>>
RMSE	Avg. RMSE	78.207 vs. 118.707	78.207 vs. 102.091
	Avg. p-value	0.00000	0.00000
	Statistical conclusion	>>	>>

6.2 EX2

In this section, we will evaluate the proposed ExOCF method and five other methods. Table 6-7, Table 6-8, Table 6-9, and Table 6-10 show the estimation accuracy of the methods across the four experiment datasets. The average estimation results of methods are shown in Figure 6-2.

The first observation from these results is that the proposed ExOCF method produces the best SSE, MdMRE, MAE, MBRE, MIBRE, RMSE, and PRED (0.25) values, suggesting that it is possible to modify the OCF method to improve its estimation accuracy. From the results obtained, we believe that applying the MLR model to the OCF variables has proven its effectiveness.

The second observation from these results is that the proposed ExOCF method improved accuracy over the baseline UCP method and other tested methods such as AOM, UCP&DT, and UCP&SVR. Table 6-11 presents the percentage improvement of the proposed ExOCF over the AOM, UCP&DT, and UCP&SVR methods averaged on all datasets. Based on this comparison, we can confidently confirm that the proposed method outperforms all other methods with superior accuracy in the evaluation criteria.

Table 6-7. Estimation results for the proposed ExOCF method and other methods on the D1 dataset

Methods	SSE	PRED	MAE	MdMRE	MBRE	MIBRE	RMSE
UCP	54838.9	0.40	95.61	0.281	0.366	0.252	104.339
OCF	14215.2	0.92	44.78	0.112	0.146	0.120	52.964
UCP&DT	2697.5	1.00	18.29	0.042	0.052	0.048	22.609
UCP&SVR	2013.1	1.00	17.12	0.048	0.049	0.046	19.588
AOM	1690.9	1.00	15.67	0.044	0.044	0.042	17.946
ExOCF	1443.2	1.00	12.61	0.029	0.035	0.033	16.392

Table 6-8. Estimation results for the proposed ExOCF method and other methods on the D2 dataset

Methods	SSE	PRED	MAE	MdMRE	MBRE	MIBRE	RMSE
UCP	34148.37	0.55	82.544	0.254	0.384	0.252	92.152
OCF	17773.29	0.65	54.393	0.165	0.227	0.167	65.596
UCP&DT	1484.69	1.00	15.068	0.043	0.049	0.045	18.782
UCP&SVR	921.93	1.00	13.213	0.035	0.043	0.041	15.137
AOM	547.19	1.00	9.647	0.029	0.031	0.030	11.691
ExOCF	501.78	1.00	8.822	0.024	0.028	0.027	11.195

Table 6-9. Estimation results for the proposed ExOCF method and other methods on the D3 dataset

Methods	SSE	PRED	MAE	MdMRE	MBRE	MIBRE	RMSE
UCP	58932.2	0.47	120.02	0.455	0.426	0.278	138.640
OCF	30148.0	0.53	87.242	0.266	0.331	0.237	95.369
UCP&DT	83.7	1.00	3.690	0.008	0.013	0.012	5.211
UCP&SVR	42.3	1.00	2.804	0.008	0.010	0.009	3.704
AOM	37.5	1.00	2.650	0.008	0.009	0.009	3.472
ExOCF	34.6	1.00	2.489	0.007	0.009	0.009	3.326

Table 6-10. Estimation results for the proposed ExOCF method and other methods on the D4 datase

Methods	SSE	PRED	MAE	MdMRE	MBRE	MIBRE	RMSE
UCP	59995.4	0.33	125.85	0.419	0.441	0.289	139.698
OCF	34210.3	0.53	87.308	0.261	0.318	0.225	98.899
UCP&DT	9867.7	1.00	21.369	0.045	0.069	0.060	29.642
UCP&SVR	10638.2	1.00	25.466	0.066	0.082	0.073	31.062
AOM	6862.6	1.00	18.481	0.034	0.058	0.053	24.829
ExOCF	5630.0	1.00	16.311	0.033	0.052	0.047	22.480

Table 6-11. The percentage improvements of the ExOCF over the other methods averaged on all datasets

Methods	SSE	MAE	RMSE	MdMRE	MBRE	MIBRE
ExOCF vs. UCP&DT	46.16%	31.13%	32.17%	32.35%	30.08%	29.97%
ExOCF vs. UCP&SVR	44.11%	31.35%	40.71%	32.28%	31.32%	23.17%
ExOCF vs. AOM	16.73%	13.39%	18.10%	13.06%	12.89%	7.84%

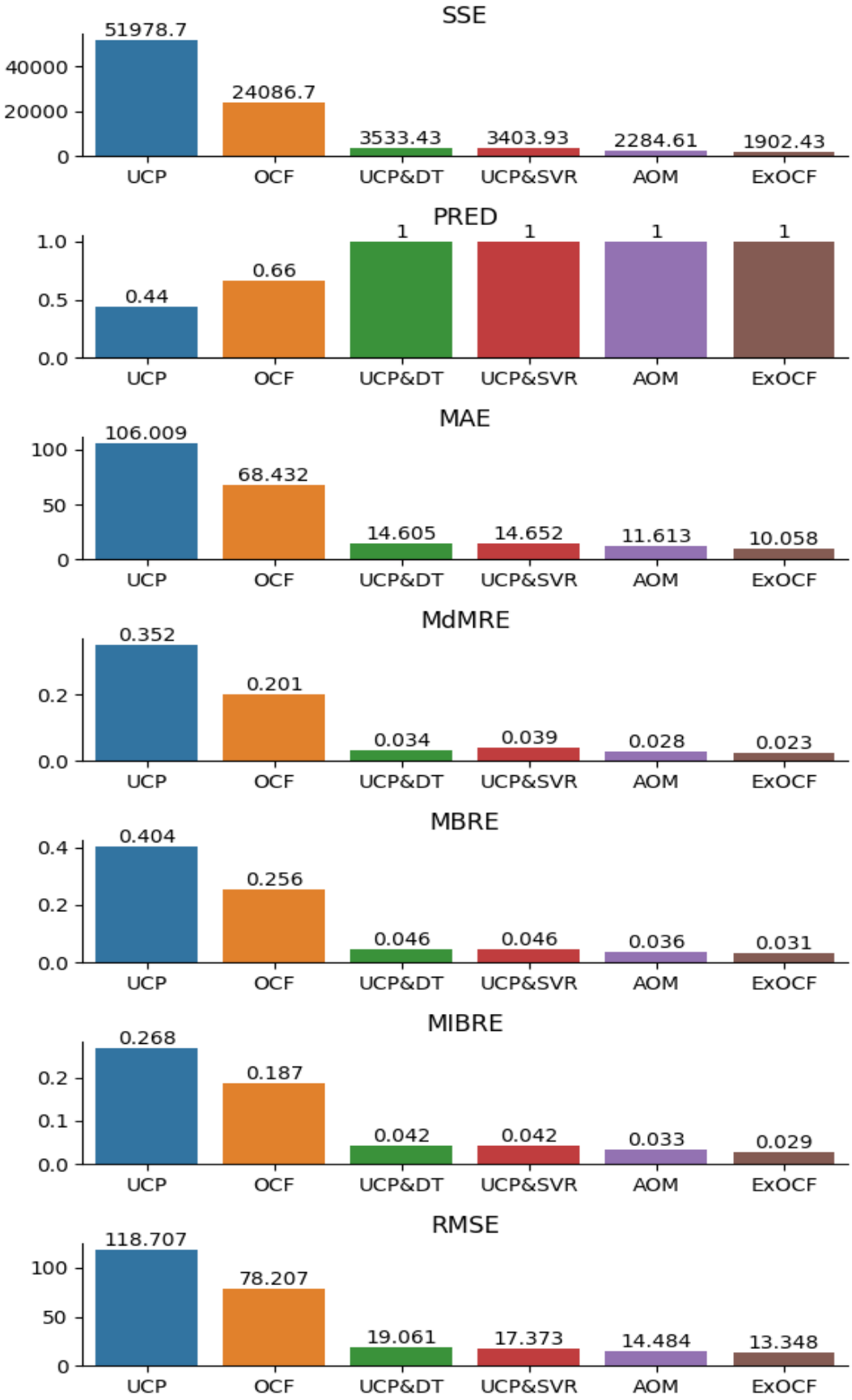


Figure 6-2. The average estimation results of the proposed ExOCF method and other methods on all datasets

Furthermore, the results confirm that the ExOCF method is statistically significant at the 95% confidence level compared to the other five methods, as shown in Table 6-12. As a result, we are inclined to accept the alternative hypothesis (H1), which is also consistent with the results presented above. A>>B means that A is statistically superior to B.

Table 6-12. The t-test results for five different runs of the proposed ExOCF method in comparison with the other methods

Pairs of methods		ExOCF vs. UCP	ExOCF vs. OCF	ExOCF vs. UCP&DT	ExOCF vs. UCP&SVR	ExOCF vs. AOM
SSE	Avg. SSE	1902.4 vs. 51,978.7	1902.4 vs. 24,086.7	1902.4 vs. 3533.4	1902.4 vs. 3403.9	1902.4 vs. 2284.6
	Avg. p-value	0.00000	0.00001	0.00267	0.00316	0.00388
	St. conc.	>>	>>	>>	>>	>>
MAE	Avg. MAE	10.058 vs. 106.009	10.058 vs. 68.432	10.058 vs. 14.605	10.058 vs. 14.651	10.058 vs. 11.613
	Avg. p-value	0.00000	0.00000	0.00001	0.00000	0.00005
	St. conc.	>>	>>	>>	>>	>>
RMSE	Avg. RMSE	13.348 vs. 118.707	13.348 vs. 78.207	13.348 vs. 19.060	13.348 vs. 17.372	13.348 vs. 14.484
	Avg. p-value	0.00000	0.00000	0.00000	0.00001	0.00007
	St. conc.	>>	>>	>>	>>	>>

6.3 EX3

Table 6-13, Table 6-14, Table 6-15, Table 6-16, Table 6-17, Table 6-18, Table 6-19, and Table 6-20 present the estimation results of the UCP-based and OCF-based single methods across the four datasets.

The first finding from these results is that the estimation accuracies of the single methods differ throughout datasets. UCP&GRNN, for example, was the best model for the D1 dataset among the UCP-based single methods. According to the

SSE, UCP&KNN had the lowest accuracy, while UCP&SVR had the most insufficient accuracy according to the MAE, RMSE, MBRE, MIBRE, and MdMRE. Similarly, among the OCF-based single methods for the D1 dataset, OCF&RF performed best according to the SSE and RMSE, while OCF&KNN performed best according to the MAE, MBRE, MIBRE, and MdMRE. The worst model was OCF&SVR.

Table 6-13. Estimation results for the UCP-based single methods on the D1 dataset

Methods	SSE	PRED	MAE	MdMRE	MBRE	MIBRE	RMSE
UCP&SVR	1866.17	0.33	16.73	0.045	0.048	0.045	18.711
UCP&MLP	1515.52	0.53	14.08	0.036	0.040	0.038	16.768
UCP&GRNN	1493.42	1.00	12.77	0.035	0.039	0.036	15.553
UCP&KNN	1942.10	1.00	16.56	0.048	0.047	0.044	18.532
UCP&DT	1520.84	1.00	13.53	0.030	0.038	0.036	16.619
UCP&RF	1526.65	1.00	14.04	0.029	0.040	0.037	16.440

Table 6-14. Estimation results for the UCP-based single methods on the D2 dataset

Methods	SSE	PRED	MAE	MdMRE	MBRE	MIBRE	RMSE
UCP&SVR	768.53	1.00	10.18	0.026	0.034	0.032	13.168
UCP&MLP	1546.82	1.00	14.85	0.040	0.050	0.046	17.333
UCP&GRNN	392.45	1.00	8.38	0.023	0.028	0.026	9.736
UCP&KNN	651.11	1.00	11.12	0.032	0.036	0.035	12.459
UCP&DT	528.28	1.00	9.497	0.027	0.031	0.030	11.151
UCP&RF	405.55	1.00	8.054	0.021	0.026	0.025	9.640

Table 6-15. Estimation results for the UCP-based single methods on the D3 dataset

Methods	SSE	PRED	MAE	MdMRE	MBRE	MIBRE	RMSE
UCP&SVR	41.978	1.00	3.066	0.014	0.011	0.010	3.573
UCP&MLP	56.090	1.00	3.629	0.015	0.012	0.012	4.050
UCP&GRNN	51.268	1.00	3.517	0.015	0.012	0.012	4.020
UCP&KNN	54.621	1.00	3.780	0.016	0.013	0.013	4.210
UCP&DT	46.617	1.00	3.305	0.013	0.011	0.011	3.767
UCP&RF	60.420	1.00	3.066	0.014	0.011	0.010	3.573

Table 6-16. Estimation results for the UCP-based single methods on the D4 dataset

Methods	SSE	PRED	MAE	MdMRE	MBRE	MIBRE	RMSE
UCP&SVR	10935.1	1.00	25.62	0.073	0.082	0.074	30.96
UCP&MLP	11890.2	0.98	25.89	0.062	0.081	0.072	31.39
UCP&GRNN	11105.5	1.00	23.82	0.056	0.076	0.067	29.95
UCP&KNN	11074.0	0.98	24.55	0.060	0.077	0.068	30.94
UCP&DT	10878.2	1.00	26.58	0.085	0.086	0.077	31.22
UCP&RF	13470.0	0.98	25.68	0.073	0.083	0.072	32.90

Table 6-17. Estimation results for the OCF-based single methods on the D1 dataset

Methods	SSE	PRED	MAE	MdMRE	MBRE	MIBRE	RMSE
OCF&SVR	1410.33	1.00	13.900	0.029	0.039	0.037	16.350
OCF&MLP	1197.73	1.00	12.887	0.031	0.036	0.035	15.362
OCF&DT	1343.01	1.00	13.470	0.030	0.038	0.036	16.021
OCF&MLR	1018.04	1.00	11.545	0.025	0.032	0.031	13.719
OCF&GB	1314.08	1.00	13.434	0.030	0.038	0.036	15.903
OCF&RF	747.09	1.00	9.520	0.022	0.027	0.026	11.700
OCF&KNN	832.11	1.00	9.245	0.017	0.026	0.024	12.356

Table 6-18. Estimation results for the OCF-based single methods on the D2 dataset

Methods	SSE	PRED	MAE	MdMRE	MBRE	MIBRE	RMSE
OCF&SVR	649.039	1.00	9.434	0.025	0.032	0.030	12.399
OCF&MLP	994.084	1.00	12.096	0.033	0.040	0.038	15.033
OCF&DT	278.476	1.00	7.203	0.022	0.023	0.023	7.949
OCF&MLR	493.827	1.00	9.479	0.026	0.031	0.029	11.017
OCF&GB	279.682	1.00	7.203	0.022	0.023	0.023	7.972
OCF&RF	360.796	1.00	7.371	0.019	0.024	0.023	9.340
OCF&KNN	244.127	1.00	6.405	0.018	0.020	0.022	7.673

Table 6-19. Estimation results for the OCF-based single methods on the D3 dataset

Methods	SSE	PRED	MAE	MdMRE	MBRE	MIBRE	RMSE
OCF&SVR	36.965	1.00	2.891	0.013	0.010	0.010	3.387
OCF&MLP	51.081	1.00	3.634	0.014	0.012	0.012	3.968
OCF&DT	37.345	1.00	2.887	0.013	0.010	0.010	3.408
OCF&MLR	46.500	1.00	3.417	0.013	0.012	0.012	3.806
OCF&GB	37.893	1.00	2.899	0.012	0.010	0.010	3.437
OCF&RF	44.123	1.00	3.132	0.013	0.011	0.011	3.700
OCF&KNN	47.462	1.00	3.312	0.014	0.011	0.011	3.830

Table 6-20. Estimation results for the OCF-based single methods on the D4 dataset

Methods	SSE	PRED	MAE	MdMRE	MBRE	MIBRE	RMSE
OCF&SVR	6642.85	1.00	18.57	0.047	0.059	0.054	23.772
OCF&MLP	6874.87	1.00	18.42	0.047	0.058	0.053	24.192
OCF&DT	10454.39	1.00	25.93	0.077	0.083	0.075	30.547
OCF&MLR	6909.89	1.00	19.05	0.054	0.060	0.054	24.246
OCF&GB	10647.20	1.00	26.23	0.084	0.085	0.076	30.810
OCF&RF	6236.12	1.00	17.97	0.050	0.056	0.051	23.387
OCF&KNN	6475.19	1.00	18.20	0.044	0.057	0.052	24.113

Furthermore, the experimental results suggest that OCF-based methods reduce estimation errors more effectively than UCP model-based methods. Table 6-21 show the percentage improvements of OCF&SVR, OCF&MLP, OCF&KNN, OCF&DT, and OCF&RF over UCP&SVR, UCP &MLP, UCP &KNN, UCP &DT, and UCP &RF. The comparison between the OCF-based and UCP-based single methods is illustrated in Figure 6-3. Based on this finding, we can conclude that approaches that use OCF variables outperform those that use UCP variables.

Table 6-21. The percentage improvements of the OCF-based single methods averaged on all datasets

Methods	SSE	MAE	RMSE	MdMRE	MBRE	MIBRE
OCF&SVR vs. UCP&SVR	35.80%	19.43%	28.17%	20.23%	19.18%	15.82%
OCF&MLP vs. UCP&MLP	39.25%	19.53%	18.50%	19.83%	18.21%	15.80%
OCF&KNN vs. UCP&KNN	44.62%	33.65%	40.18%	34.06%	32.00%	27.47%
OCF&DT vs. UCP&DT	6.63%	6.49%	9.28%	7.23%	6.52%	7.70%
OCF&RF vs. UCP&RF	52.22%	26.56%	25.50%	27.71%	25.30%	24.08%

Table 6-22 and Table 6-23 present the ranking the UCP-based and OCF-based single methods across the datasets using the SSE criterion. The "1" represents the best method and "6" or "7" indicates the worst method. Based on these results, we can conclude that there is no best method, indicating that a single model can outperform in one dataset while underperforming in another.

Table 6-22. Rank the UCP-based single approaches from 1 to 6 based on the SSE metric

Methods	D1	D2	D3	D4
UCP&SVR	5	5	1	2
UCP&MLP	2	6	5	5
UCP&GRNN	1	1	3	4
UCP&KNN	6	4	4	3
UCP&DT	3	3	2	1
UCP&RF	4	2	6	6

Table 6-23. Rank the UCP-based single approaches from 1 to 7 based on the SSE metric

Methods	D1	D2	D3	D4
OCF&SVR	7	6	1	3
OCF&MLP	4	7	7	4
OCF&DT	6	2	2	6
OCF&MLR	3	5	5	5
OCF&GB	5	3	3	7
OCF&RF	1	4	4	1
OCF&KNN	2	1	6	2

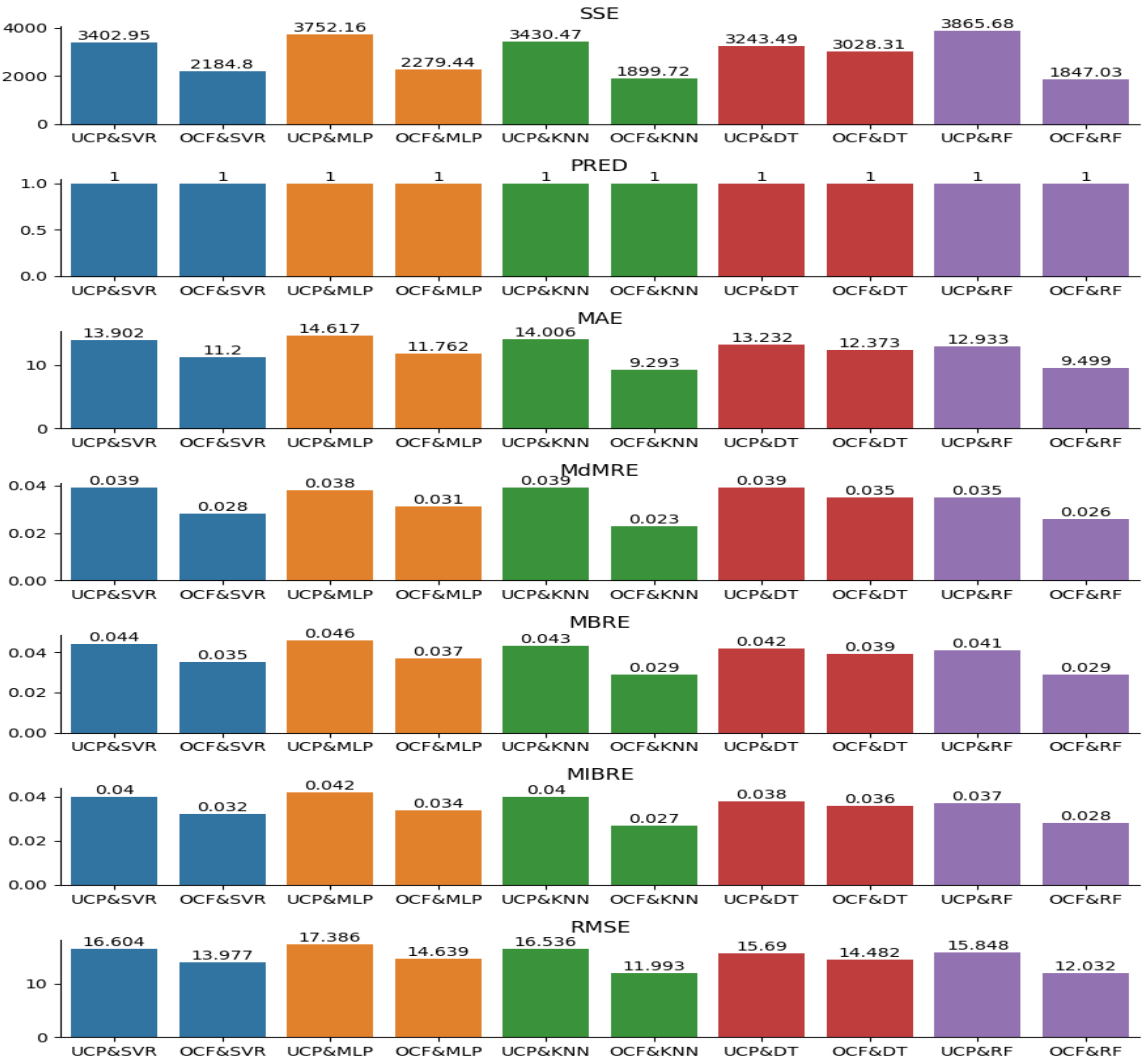


Figure 6-3. The average estimation results of the UCP-based and OCF-based single methods on all datasets

The experimental results for the two ensemble methods (VUCP and SOCF) and their single approaches are shown in Table 6-24, Table 6-25, Table 6-26, and Table 6-27. The comparison between between the ensemble methods and their single approaches is shown in Figure 6-4 and Figure 6-5.

Table 6-24. Estimation results for the ensemble methods on the D1 dataset

Methods	SSE	PRED	MAE	MdMRE	MBRE	MIBRE	RMSE
VUCP	1173.227	1.00	11.575	0.027	0.033	0.031	13.970
SOCF	491.627	1.00	7.168	0.016	0.020	0.023	9.186

Table 6-25. Estimation results for the ensemble methods on the D2 dataset

Methods	SSE	PRED	MAE	MdMRE	MBRE	MIBRE	RMSE
VUCP	335.898	1.00	7.618	0.023	0.025	0.024	8.937
SOCF	125.236	1.00	4.322	0.013	0.014	0.022	5.386

Table 6-26. Estimation results for the ensemble methods on the D3 dataset

Methods	SSE	PRED	MAE	MdMRE	MBRE	MIBRE	RMSE
VUCP	38.537	1.00	2.865	0.013	0.010	0.010	3.431
SOCF	31.496	1.00	2.486	0.008	0.009	0.010	3.106

Table 6-27. Estimation results for the ensemble methods on the D4 dataset

Methods	SSE	PRED	MAE	MdMRE	MBRE	MIBRE	RMSE
VUCP	8043.400	1.00	21.536	0.059	0.069	0.062	26.338
SOCF	4222.464	1.00	13.944	0.030	0.043	0.049	21.706

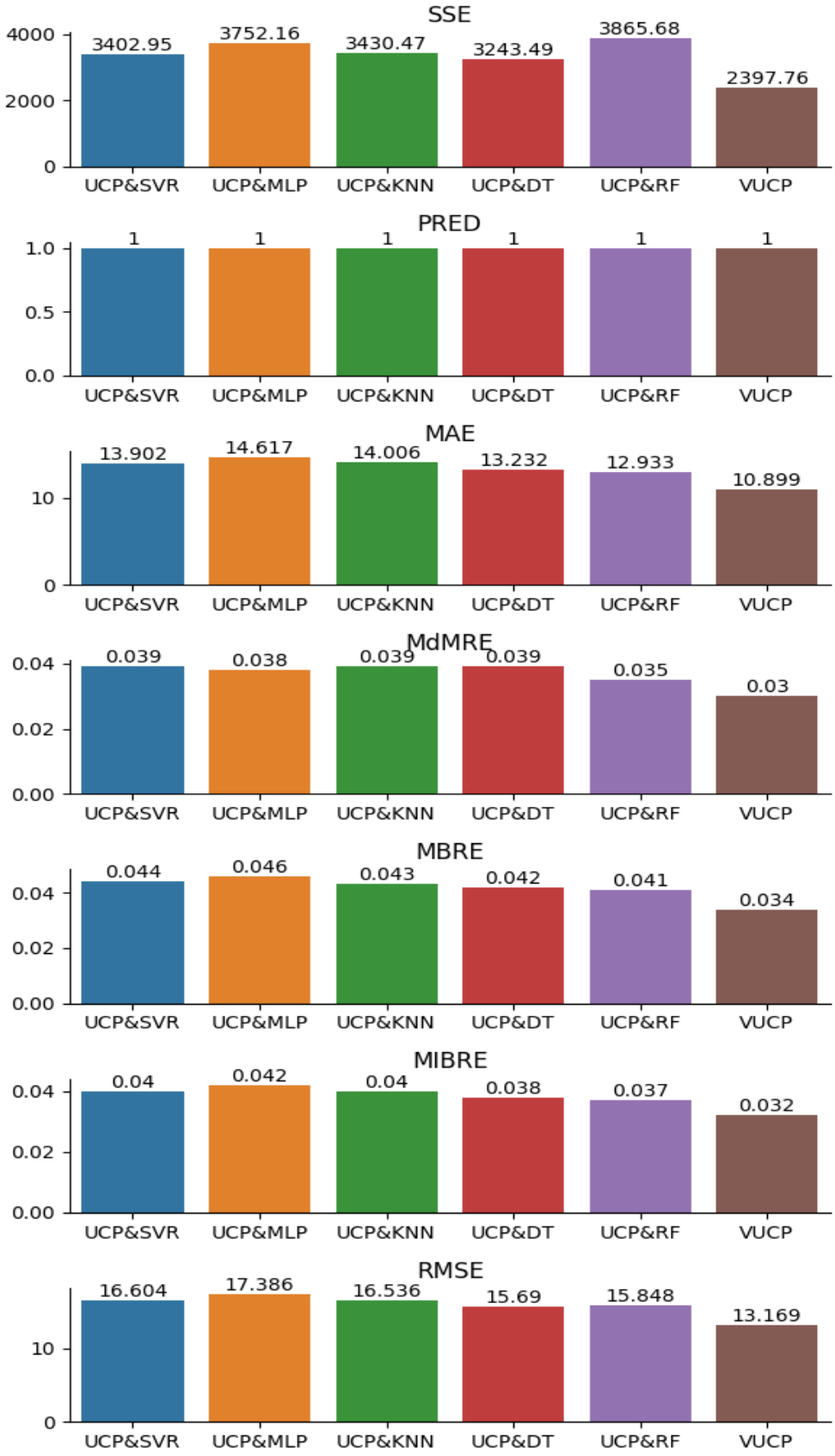


Figure 6-4. The comparison between the ensemble method VUCP and its single approaches

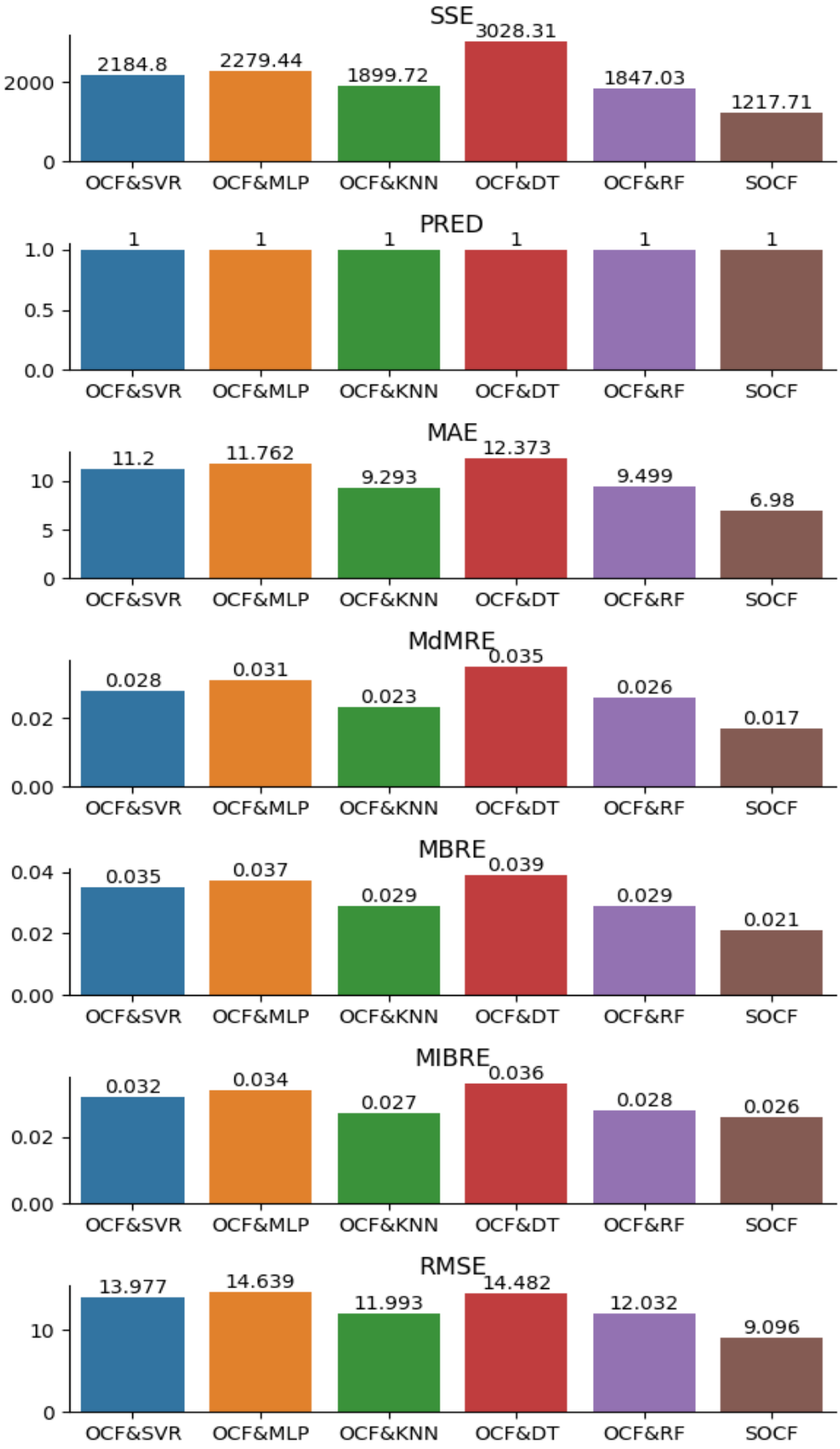


Figure 6-5. The comparison between the ensemble method SOCF and its single approaches

Based on these results, we can conclude that the ensemble methods outperform their single methods, and the proposed SOCF approach surpasses the VUCP method. Moreover, the results confirm that the SOCF method is statistically significant at the 95% confidence level compared to the other methods, as shown Table 6-28, Table 6-29, and Table 6-30. A>> B means that A is statistically superior to B. Therefore, we accept the alternative hypothesis H1.

Table 6-28. The t-test results for five different runs of the proposed SOCF method in comparison with the other methods

Pairs of methods		SOCF vs. UCP	SOCF vs. OCF&MLP	SOCF vs. OCF&DT	SOCF vs. OCF&SVR	SOCF vs. OCF&MLR
SSE	Avg. SSE	1217.7 vs. 54838.9	1217.71 vs. 2279.44	1217.71 vs. 3028.31	1217.71 vs. 2184.80	1217.71 vs. 2117.07
	Avg. value	0.00000	0.00076	0.00440	0.00190	0.00514
	St. conc.	>>	>>	>>	>>	>>
MAE	Avg. MAE	6.980 vs. 95.615	6.980 vs. 11.762	6.980 vs. 12.373	6.980 vs. 11.200	6.980 vs. 10.874
	Avg. value	0.00000	0.00000	0.00005	0.00000	0.00001
	St. conc.	>>	>>	>>	>>	>>
RMSE	Avg. RMSE	9.096 vs. 104.339	9.096 vs. 14.639	9.096 vs. 14.482	9.096 vs. 13.977	9.096 vs. 13.197
	Avg. value	0.00000	0.00002	0.00006	0.00005	0.00000
	St. conc.	>>	>>	>>	>>	>>

Table 6-29. The t-test results for five different runs of the proposed SOCF method in comparison with the other methods

Pairs of methods		SOCF vs. OCF&GB	SOCF vs. OCF&RF	SOCF vs. UCP&KNN	SOCF vs. UCP&SVR	SOCF vs. UCP&MLP
SSE	Avg. SSE	1217.7 vs. 3069.7	1217.7 vs. 1847.0	1217.71 vs. 1899.03	1217.71 vs. 3402.95	1217.7 vs. 2117.07
	Avg. value	0.00460	0.00583	0.01199	0.00195	0.00514
	St. conc.	>>	>>	>>	>>	>>
MAE	Avg. MAE	6.980 vs. 12.441	6.980 vs. 9.499	6.980 vs. 9.293	6.980 vs. 13.902	6.980 vs. 10.874
	Avg. value	0.00005	0.00000	0.00005	0.00005	0.00001
	St. conc.	>>	>>	>>	>>	>>
RMSE	Avg. RMSE	9.096 vs. 14.530	9.096 vs. 12.032	9.096 vs. 11.993	9.096 vs. 16.604	9.096 vs.
	Avg. value	0.00006	0.00005	0.00010	0.00000	13.197
	St. conc.	>>	>>	>>	>>	>>

Table 6-30. The t-test results for five different runs of the proposed SOCF method in comparison with the other methods

Pairs of methods		SOCF vs. UCP&GB	SOCF vs. OCF&KN	SOCF vs. UCP&DT	SOCF vs. UCP&RF	SOCF vs. VUCP
SSE	Avg. SSE	1217.7 vs. 3069.7	1217.7 vs. 1847.0	1217.71 vs. 1899.03	1217.71 vs. 3402.95	1217.71 vs. 2397.77
	Avg. value	0.00460	0.00583	0.01199	0.00195	0.00764
	St. conc.	>>	>>	>>	>>	>>
MAE	Avg. MAE	6.980 vs. 12.441	6.980 vs. 9.499	6.980 vs. 9.293	6.980 vs. 13.902	6.980 vs. 10.899
	Avg. value	0.00005	0.00000	0.00005	0.00005	0.00003
	St. conc.	>>	>>	>>	>>	>>
RMSE	Avg. RMSE	9.096 vs. 14.530	9.096 vs. 12.032	9.096 vs. 11.993	9.096 vs. 16.604	9.096 vs. 13.169
	Avg. value	0.00006	0.00005	0.00010	0.00000	0.00007
	St. conc.	>>	>>	>>	>>	>>

Besides that, we also performed ablation analyses to evaluate the effectiveness of each of SOCF's three core components. Table 6-31 and Table 6-32 show that the average SSE, MAE, and RMSE results for SOCF increased when the three components were replaced from the model, implying a decrease in estimation accuracy in each case. The ↑ sign denotes an increase in SSE, MAE, RMSE, MBRE, MIBRE, or MdMRE results, implying a decrease in estimation accuracy compared to the SOCF (in Full) model. The term "<< Full SOCF model" refers to the full SOCF model's statistical superiority over models that exclude one of the three core components.

- SOCF-Case1: Removing the first component (optimizing model parameters using the GS technique) and using the default parameters for SOCF's single methods.
- SOCF-Case2: Removing the second component (reducing generalization error using the stacking ensemble) and using the voting ensemble.

- SOCF-Case3: Removing the third component (the selection of seven single methods) and using three methods: MLR, SVR, and MLP.

Table 6-31. The results for SOCF-Case1, SOCF-Case2, and SOCF-Case3

Methods	SSE	MAE	RMSE
SOCF (in Full)	1217.7	6.98	9.10
SOCF-Case1	↑1412.80	↑8.10	↑10.31
SOCF-Case2	↑1643.75	↑8.91	↑11.16
SOCF-Case3	↑2146.13	↑11.00	↑14.03

Table 6-32. The ablation analyses for SOCF-Case1, SOCF-Case2, and SOCF-Case3

Models for ablation analyses			p-value of t-test
SOCF-Case1	SSE increase	195.096	0.01166 << Full SOCF model
	MAE increase	1.119	0.00000 << Full SOCF model
	RMSE increase	1.215	0.00008 << Full SOCF model
SOCF-Case2	SSE increase	426.047	0.01914 << Full SOCF model
	MAE increase	1.932	0.00035 << Full SOCF model
	RMSE increase	2.061	0.00011 << Full SOCF model
SOCF-Case3	SSE increase	928.428	0.00070 << Full SOCF model
	MAE increase	4.029	0.00001 << Full SOCF model
	RMSE increase	4.930	0.00001 << Full SOCF model

6.4 EX4

In the EX4, we will compare the proposed OCF(PFCFE) method as well as the previous related methods (UCP, SW, and OCF) based on the four experimental datasets. The obtained results from Table 6-33, Table 6-34, Table 6-35, and Table 6-36 allow us to confidently conclude that the OCF(PFCFE) using the proposed software productivity approach achieves better improvements than the previous related methods using fixed productivity metrics, concerning all accuracy measures. The comparison between the OCF(PFCFE) method and three related methods is illustrated in Figure 6-6.

The percentage improvements of the proposed OCF(PFCFE) over the other methods are presented in Table 6-37. This conclusion is confirmed by statistical t-test comparisons for each corresponding method (see Table 6-38). A>>B refers to A statistical superiority to B. The OCF(PFCFE) using the proposed software productivity approach is statistically better than other methods, as the obtained p-values are all below 0.05.

Table 6-33. Estimation results for the proposed OCF(PF_{CFE}) method and other methods on the D1 dataset

Methods	SSE	PRED	MAE	MdMRE	MBRE	MIBRE	RMSE
UCP	2.77E+07	0.4	1,884.6	0.280	0.418	0.258	2,129.9
SW	2.20E+07	0.5	1,642.9	0.239	0.328	0.222	1,909.8
OCF	1.73E+07	0.7	1,607.9	0.226	0.302	0.223	1,691.8
OCF (PF_{CFE})	6.05E+06	0.8	804.0	0.079	0.113	0.097	979.9

Table 6-34. Estimation results for the proposed OCF(PF_{CFE}) method and other methods on the D2 dataset

Methods	SSE	PRED	MAE	MdMRE	MBRE	MIBRE	RMSE
UCP	6.95E+07	0.32	2,920.5	0.364	0.655	0.345	3,721.7
SW	6.70E+07	0.32	2,847.4	0.364	0.593	0.334	3,656.4
OCF	2.61E+07	0.36	1,986.6	0.304	0.559	0.302	2,264.1
OCF (PF_{CFE})	2.14E+07	0.64	1,591.8	0.190	0.302	0.202	2,056.2

Table 6-35. Estimation results for the proposed OCF(PF_{CFE}) method and other methods on the D3 dataset

Methods	SSE	PRED	MAE	MdMRE	MBRE	MIBRE	RMSE
UCP	3.09E+07	0.37	1,939.0	0.313	0.383	0.250	2,262.8
SW	5.43E+07	0.37	2,551.3	0.450	0.446	0.283	3,004.1
OCF	2.97E+07	0.33	1,931.5	0.300	0.363	0.247	2,219.3
OCF (PF_{CFE})	1.13E+07	0.73	1,061.4	0.130	0.244	0.164	1,352.6

Table 6-36. Estimation results for the proposed OCF(PF_{CFE}) method and other methods on the D4 dataset

Methods	SSE	PRED	MAE	MdMRE	MBRE	MIBRE	RMSE
UCP	8.05E+07	0.53	1,940.6	0.255	0.413	0.252	2,367.7
SW	8.54E+07	0.61	1,872.8	0.242	0.356	0.226	2,423.6
OCF	5.27E+07	0.61	1,566.9	0.215	0.350	0.220	1,922.7
OCF (PF_{CFE})	4.78E+07	0.69	1,430.1	0.186	0.255	0.182	1,827.7

Table 6-37. The percentage improvements of the proposed OCF(PF_{CFE}) method averaged on all datasets

Methods	SSE	PRED	MAE	RMSE	MdMRE	MBRE	MIBRE
OCF(PF_{CFE}) vs. UCP	58.6%	43.4%	43.7%	40.7%	51.7%	51.1%	41.6%
OCF(PF_{CFE}) vs. SW	62.2%	37.1%	45.2%	43.5%	54.8%	47.0%	39.4%
OCF(PF_{CFE}) vs. OCF	31.3%	30.1%	31.1%	23.2%	44.0%	41.9%	35.0%
OCF(PF_{CFE}) vs. AOM	58.6%	43.4%	43.7%	40.7%	51.7%	51.1%	41.6%
OCF(PF_{CFE}) vs. UCP	62.2%	37.1%	45.2%	43.5%	54.8%	47.0%	39.4%

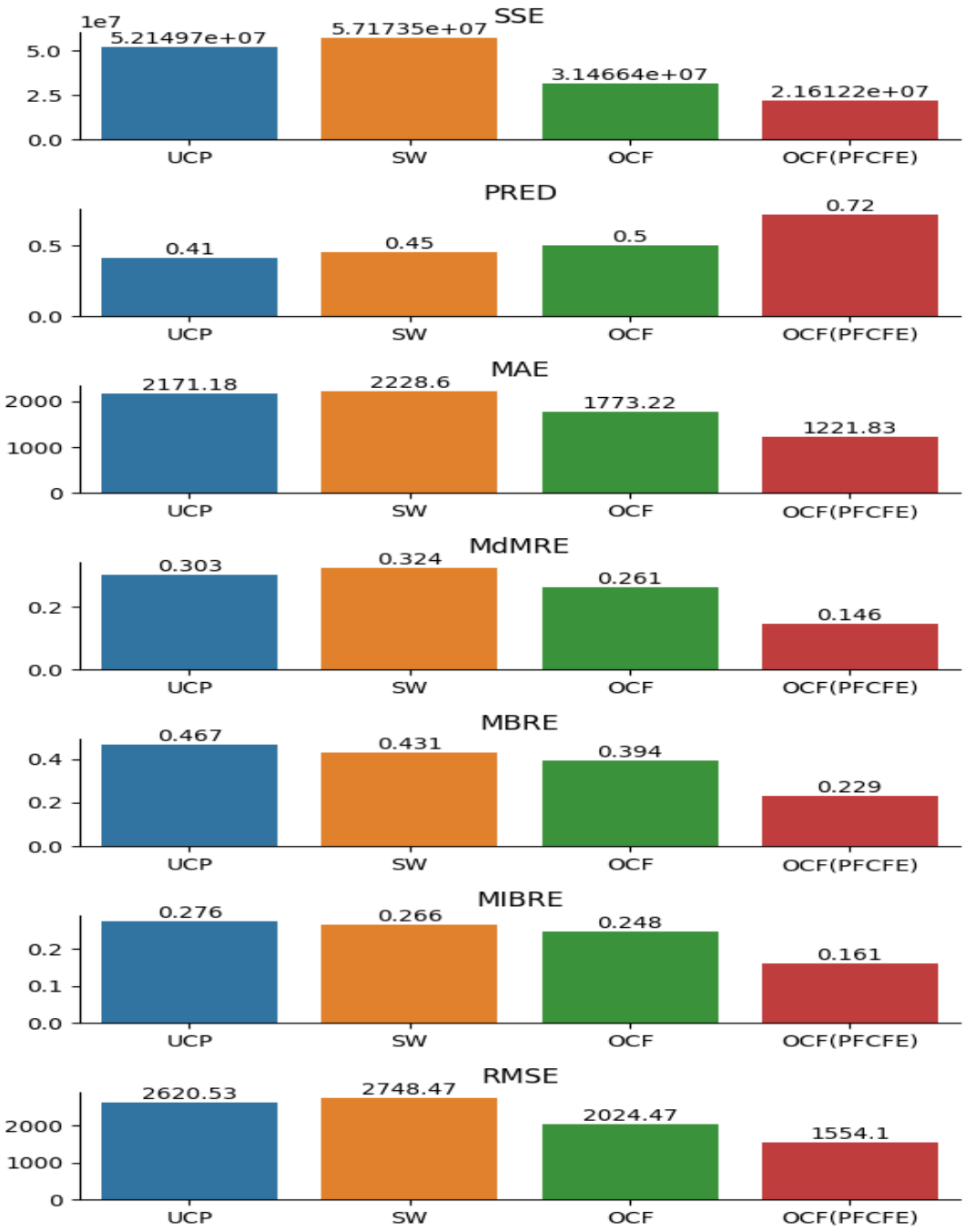


Figure 6-6. The average estimation results of the proposed OCF(PF_{CFE}) method and other methods on all dataset

Table 6-38 The t-test results of five different runs for statistical comparison of our proposed OCF(PF_{CFE}) methods with other tested methods

Pairs of methods		OCF(PF_{CFE}) vs. UCP	OCF(PF_{CFE}) vs. SW	OCF(PF_{CFE}) vs. OCF	OCF(PF_{CFE}) vs. AOM
SSE	Avg. SSE	2.16E+07 vs. 5.21E+07	2.16E+07 vs. 5.72E+07	2.16E+07 vs. 3.15E+07	2.16E+07 vs. 3.75E+07
	Avg. p-value	0.00000	0.00000	0.00000	0.00011
	St. conc.	>>	>>	>>	>>
MAE	Avg. MAE	1194.141 vs. 2171.213	1194.141 vs. 2228.636	1194.141 vs. 1773.276	1194.141 vs. 1756.148
	Avg. p-value	0.00000	0.00000	0.00000	0.00000
	St. conc.	>>	>>	>>	>>
RMSE	Avg. RMSE	1554.136 vs. 2620.587	1554.136 vs. 2748.516	1554.136 vs. 2024.497	1554.136 vs. 2126.666
	Avg. p-value	0.00000	0.00000	0.00000	0.00000
	St. conc.	>>	>>	>>	>>

7 THREAT OF VALIDITY

Threats to the validity of this study, namely to internal, construct, and external validity, are summarized as follows:

Internal validity: In EX1, there is no better way to choose the regularization parameter λ to extract a specific set of variables when using LASSO regression, as shown in equation (4.2). Controlling for the strength of the penalty (tuning λ) has a significant impact. For example, if λ is sufficiently large, the coefficients must be precisely zero, reducing dimensionality. The larger the parameter λ, the larger the number of coefficients reduced to zero. Therefore, we determined the λ-value using the LOOCV method, where the R-squared reaches its highest value. In addition, the LOOCV method was used to determine the optimal configuration parameters for the statistical and ML algorithms in EX4. The unbiased performance evaluation methodology of each statistical and ML algorithm should correct any overfitting of the proposed methods [78], [79]. The LOOCV approach is a better evaluation method than cross-validation because it provides lower bias, greater variance estimation, and adaptability for small datasets. Each ML technique uses the GS technique to optimize configuration parameters. Adding a tuning phase would significantly increase the cost of the experiments, and most approaches in our study worked well with optimally configured parameter values. However, these settings may not work well on larger datasets. Besides that, the dataset was collected over a long period by three donors. The data providers provided some independent variables. The process of using case point calculation, particularly for factor weights, is unknown. This may affect data quality and comparability across data providers. Previous articles used preprocessed datasets, which may have affected reliability.

Construct validity is related to the generalizability of the results. The goal of this study was to reduce estimation error. The process is based on a standard procedure for tuning an estimation model. Performing a 5-fold cross-validation and processing four datasets allow us to generalize the results. To eliminate monomethod bias, unbiased evaluation criteria such as SSE, PRED (0.25), MAE, MdMRE, MBRE, MIBRE, and RMSE, as well as statistical pairwise t-tests, were used to determine the validity of the results. Therefore, we can conclude that the experimental results of this work are highly generalizable.

External validity: The first one is the experimental dataset. Since our studies are based on publicly available datasets, the results should be convincing. These datasets contain a small fraction of all datasets in the real world. Consequently, conclusions from these datasets may not be comparable to other datasets. The second point concerns the use of the GS technique to fine-tune the configuration settings of each statistical and ML approach. It is recommended that numerous optimization approaches be explored to generalize the results of this study.

8 CONTRIBUTIONS OF THE THESIS TO SCIENCE AND PRACTICE

The main benefit of this work is the introduction of a new approach to complex algorithms based on engineering requirements research for a more accurate estimation of software effort. The new algorithms are inspired by the possibilities of using a standardized estimation procedure to address the impact of human error in UCM analysis and to simplify the original UCP principles.

The main benefits of this work can be summarized as follows:

- Proposed procedures can help project managers reduce risks in evaluating correction factors and obtain effort estimates.
- An algorithm for calculating productivity based on correction factors has been proposed through a voting set approach consisting of three ML techniques.
- Proposed a comprehensive approach to improve estimation accuracy and minimize project risks in the early stages of software development.
- Experiments have shown that the use of the proposed new algorithms minimizes the estimation error compared to the selected methods.

In summary, the results obtained can be considered beneficial for industrial applications, as they show that the proposed algorithms lead to more accurate estimates of the size and complexity of the software.

9 CONCLUSIONS

The presented doctoral thesis is proposed UCP-based estimation methods in the early stages of software development. Our methods can help project managers estimate costs early and efficiently, avoiding project overestimation and late delivery, among other issues. Each approach has its advantages, and they complement each other to form a complete process and promote significant efficiency to minimize estimation error more efficiently in all situations. The results show that our proposed SDEE method outperforms other related methods.

One of our future works is to calibrate the weighting values of the correction factors to reflect the latest trend in the software development industry and improve the accuracy of the proposed methods. Therefore, an approach to calibrate the weights of the correction factors using an artificial neural network will be performed in the future. Another concern relates to a key aspect of the heterogeneity of the historical data. This could lead to an increase in the estimation error for SDEE. The use of clustering approaches is considered a solution to improve the method's estimation accuracy in our future work.

10 LITERATURE

[1] B. W. Boehm, "Software Engineering Economics," *IEEE Transactions on Software Engineering,* vol. SE-10, *(1),* 1984.

[2] B. Boehm *et al*, "Software Cost Estimation with COCOMO II. Prentice Hall," *Upper Saddle River, NJ,* 2000.

[3] M. Jørgensen and M. Shepperd, "A systematic review of software development cost estimation studies," *IEEE Transactions on Software Engineering,* vol. 33, *(1),* 2007.

[4] A. Trendowicz, J. Münch and R. Jeffery, "State of the practice in software effort estimation: A survey and literature review," in *IFIP Central and East European Conference on Software Engineering Techniques,* pp. 232-245, 2008.

[5] Nhung, Ho Le Thi Kim, H. T. Hoc and V. V. Hai, "A review of use case-based development effort estimation methods in the system development context," *Proceedings of the Computational Methods in Systems and Software,* pp. 484-499, 2019.

[6] B. Boehm, C. Abts and S. Chulani, "Software development cost estimation approaches — A survey," *Annals of Software Engineering,* vol. 10, *(1/4),* pp. 177-205, 2000.

[7] R. N. Charette, "Why Software Fails," *IEEE Spectrum,* vol. 42, *(9),* 2005.

[8] Arlene Minkiewicz, "Use Case Sizing," *PRICE Systems, L.L.C,* 2015.

[9] C. J. Neill and P. A. Laplante, "Requirements Engineering: The State of the Practice," *IEEE Software,* vol. 20, *(6),* 2003.

[10] Gustav Karner, "Resource Estimation for Objector Projects," 1993.

[11] M. Azzeh, A. Bou Nassif and I. B. Attili, "Predicting software effort from use case points: A systematic review," *Science of Computer Programming,* vol. 204, 2021.

[12] V. Khatibi and D. N. a. Jawawi, "Software Cost Estimation Methods : A Review," *Journal of Emerging Trends in Computing and Information Sciences,* vol. 2, *(1),* 2010.

[13] B. Marapelli, A. Carie and S. M. Islam, "Software effort estimation with use case points using ensemble machine learning models," in *International Conference on Electrical, Computer and Energy Technologies (ICECET),* 2021.

[14] R. Silhavy, P. Silhavy and Z. Prokopova, "Using actors and use cases for software size estimation," *Electronics (Switzerland),* vol. 10, *(5),* 2021.

[15] M. Manzoor and A. Wahid, "Revised Use Case Point (Re-UCP) Model for Software Effort Estimation," *International Journal of Advanced Computer Science and Applications,* vol. 6, *(3),* 2015.

[16] F. Wang *et al*, "Extended use case points method for software cost estimation," in *International Conference on Computational Intelligence and Software Engineering,* 2009.

[17] K. Periyasamy and A. Ghode, "Cost estimation using extended use case point (e-UCP) model," in *2009 International Conference on Computational Intelligence and Software Engineering,* 2009.

[18] M. Jørgensen, "Regression models of software development effort estimation accuracy and bias," *Empirical Software Engineering,* vol. 9, *(4),* 2004.

[19] S. Humpage, "An introduction to regression analysis," *Sensors (Peterborough, NH),* vol. 17, *(9),* 2000.

[20] V. Khatibi Bardsiri *et al*, "A flexible method to estimate the software development effort based on the classification of projects and localization of comparisons," *Empirical Software Engineering,* vol. 19, *(4),* 2014.

[21] M. Shepperd and S. MacDonell, "Evaluating prediction systems in software project estimation," *Information and Software Technology,* vol. 54, *(8),* 2012.

[22] M. Azzeh *et al*, "Pareto efficient multi-objective optimization for local tuning of analogy-based estimation," *Neural Computing and Applications,* vol. 27, *(8),* 2016.

[23] R. Silhavy, P. Silhavy and Z. Prokopova, "Algorithmic optimisation method for improving use case points estimation," *PLoS ONE,* vol. 10, *(11),* 2015.

[24] N. Nunes, L. Constantine and R. Kazman, "IUCP: Estimating interactive-software project size with enhanced use-case points," *IEEE Software,* vol. 28, *(4),* 2011.

[25] A. R. Gray and S. G. MacDonell, "A comparison of techniques for developing predictive models of software metrics," *Information and Software Technology,* vol. 39, *(6),* 1997.

[26] R. Alves, P. Valente and N. J. Nunes, "Improving software effort estimation with human-centric models: A comparison of UCP and iUCP accuracy," in *EICS 2013 - Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems,* 2013.

[27] P. Jovan *et al*, "Enhancing use case point estimation method using fuzzy algorithms," in *2015 23rd Telecommunications Forum Telfor (TELFOR),* 2015.

[28] M. Saroha and S. Sahu, "Software effort estimation using enhanced use case point model," in *Nternational Conference on Computing, Communication and Automation, ICCCA 2015,* 2015.

[29] L. M. Huanca and S. B. Oré, "Factors affecting the accuracy of use case points," in *International Conference on Software Process Improvement,* 2016.

[30] A. B. Nassif, D. Ho and L. F. Capretz, "Towards an early software estimation using log-linear regression and a multilayer perceptron model," *Journal of Systems and Software,* vol. 86, *(1),* 2013.

[31] Sholiq, R. S. Dewi and A. P. Subriadi, "A comparative study of software development size estimation method: UCPabc vs function points," in *Procedia Computer Science,* 2017.

[32] P. Mohagheghi, B. Anda and R. Conradi, "Effort estimation of use cases for incremental large-scale software development," in *Proceedings - 27th International Conference on Software Engineering, ICSE05,* 2005.

[33] M. R. Braz and S. R. Vergilio, "Software effort estimation based on use cases," in *Proceedings - International Computer Software and Applications Conference,* 2006.

[34] K. Qi *et al*, "Calibrating use case points using bayesian analysis," in *International Symposium on Empirical Software Engineering and Measurement,* 2018.

[35] K. Rak, Ž Car and I. Lovrek, "Effort estimation model for software development projects based on use case reuse," *Journal of Software: Evolution and Process,* vol. 31, *(2),* 2019.

[36] G. Robiolo, C. Badano and R. Orosco, "Transactions and paths: Two use case based metrics which improve the early effort estimation," in *3rd International Symposium on Empirical Software Engineering and Measurement, ESEM 2009,* 2009.

[37] L. Lavazza and G. Robiolo, "The role of the measure of functional complexity in effort estimation," in *ACM International Conference Proceeding Series,* 2010.

[38] P. S. Kumar *et al*, "Advancement from neural networks to deep learning in software effort estimation: Perspective of two decades," *Computer Science Review,* vol. 38, 2020.

[39] T. M Kiran Kumar and M. A. Jayaram, "Comparison of hard limiting and soft computing methods for predicting software effort estimation: In reference to Small Scale Visualization Projects," *International Journal of Engineering & Technology,* vol. 7, *(4.6),* 2018.

[40] Jose Thiago, Jose Thiago H. and A. L. I. Oliveira, "Ensemble Effort Estimation using dynamic selection," *Journal of Systems and Software,* vol. 175, 2021.

[41] A. G. Priya Varshini, K. Anitha Kumari and V. Varadarajan, "Estimating software development efforts using a random forest-based stacked ensemble approach," *Electronics (Switzerland),* vol. 10, *(10),* 2021.

[42] M. Jørgensen, U. Indahl and D. Sjøberg, "Software effort estimation by analogy and "regression toward the mean"," in *Journal of Systems and Software,* 2003.

[43] J. Heidrich, M. Oivo and A. Jedlitschka, "Software productivity and effort estimation," *Journal of Software: Evolution and Process,* vol. 27, *(7),* 2015.

[44] D. Rodríguez *et al*, "Empirical findings on team size and productivity in software development," *Journal of Systems and Software,* vol. 85, *(3),* 2012.

[45] K. Petersen, "Measuring and predicting software productivity: A systematic map and review," *Information and Software Technology,* vol. 53, *(4),* 2011.

[46] B. Kitchenham and E. Mendes, "Software productivity measurement using multiple size measures," *IEEE Transactions on Software Engineering,* vol. 30, *(12),* 2004.

[47] L. M. Alves *et al*, "An empirical study on the estimation of software development effort with use case points," in *Proceedings - Frontiers in Education Conference, FIE,* 2013.

[48] M. Azzeh, A. B. Nassif and L. L. Minku, "An empirical evaluation of ensemble adjustment methods for analogy-based effort estimation," in *Journal of Systems and Software,* 2015.

[49] N. A. Zakaria *et al*, "Software Project Estimation with Machine Learning," *International Journal of Advanced Computer Science and Applications,* vol. 12, *(6),* 2021.

[50] A. A. Abdulmajeed, M. A. Al-Jawaherry and T. M. Tawfeeq, "Predict the required cost to develop software engineering projects by using machine learning," in *Journal of Physics: Conference Series,* 2021.

[51] S. Shukla and S. Kumar, "An extreme learning machine based approach for software effort estimation." in *ENASE 2021 - 16th International Conference on Evaluation of Novel Approaches to Software Engineering,* 2021.

[52] A. J. Singh and M. Kumar, "Comparative Analysis on Prediction of Software Effort Estimation Using Machine Learning Techniques," *SSRN Electronic Journal,* 2020.

[53] B. Marapelli and P. Peddi, "Software Development Effort Duration and Cost Estimation using Linear Regression and K-Nearest Neighbors Machine Learning Algorithms," *International Journal of Innovative Technology and Exploring Engineering,* vol. 9, *(2),* pp. 1043-1047, 2019.

[54] S. Shukla and S. Kumar, "Applicability of neural network based models for software effort estimation," in *2019 IEEE World Congress on Services (SERVICES),* 2019.

[55] M. Azzeh, A. B. Nassif and S. Banitaan, "Comparative analysis of soft computing techniques for predicting software effort based use case points," *IET Software,* vol. 12, *(1),* 2018.

[56] H. Mustapha and N. Abdelwahed, "Investigating the use of random forest in software effort estimation," *Procedia Computer Science,* vol. 148, pp. 343-352, 2019.

[57] A. García-Floriano *et al*, "Support vector regression for predicting software enhancement effort," *Information and Software Technology,* vol. 97, 2018.

[58] A. Banimustafa, "Predicting software effort estimation using machine learning techniques," in *8th International Conference on Computer Science and Information Technology, CSIT 2018,* 2018.

[59] P. Sharma and J. Singh, "Machine learning based effort estimation using standardization," in *2018 International Conference on Computing, Power and Communication Technologies (GUCON),* 2018.

[60] O. Hidmi and B. E. Sakar, "Software Development Effort Estimation Using Ensemble Machine Learning," *International Journal of Computing, Communication and Instrumentation Engineering,* vol. 4, *(1),* 2017.

[61] S. M. Satapathy and S. K. Rath, "Empirical assessment of machine learning models for agile software development effort estimation using story points," *Innovations in Systems and Software Engineering,* vol. 13, *(2-3),* 2017.

[62] R. Silhavy, P. Silhavy and Z. Prokopova, "Analysis and selection of a regression model for the Use Case Points method using a stepwise approach," *Journal of Systems and Software,* vol. 125, 2017.

[63] S. M. Satapathy, B. P. Acharya and S. K. Rath, "Early stage software effort estimation using random forest technique based on use case points," *IET Software,* vol. 10, *(1),* 2016.

[64] Z. Tao *et al*, "GA-SVM based feature selection and parameter optimization in hospitalization expense modeling," *Applied Soft Computing Journal,* vol. 75, 2019.

[65] R. Tadeusiewicz, "Neural networks: A comprehensive foundation," *Control Engineering Practice,* vol. 3, *(5),* 1995.

[66] M. J. a. Berry and G. S. Linoff, *Data Mining Techniques: For Marketing, Sales, and Customer Relationship Management.* 2004.

[67] C. Cortes and V. Vapnik, "Support-Vector Networks," *Machine Learning,* vol. 20, *(3),* 1995.

[68] A. Najm, A. Zakrani and A. Marzak, "Decision trees based software development effort estimation: A systematic mapping study," in *Proceedings of 2019 International Conference of Computer Science and Renewable Energies, ICCSRE 2019,* 2019.

[69] S. L. Salzberg, "C4.5: Programs for Machine Learning by J. Ross Quinlan. Morgan Kaufmann Publishers, Inc., 1993," *Machine Learning,* vol. 16, *(3),* 1994.

[70] E. R. Ziegel, "The Elements of Statistical Learning," *Technometrics,* vol. 45, *(3),* 2003.

[71] L. Breiman, "Random forests," *Machine Learning,* vol. 45, *(1),* 2001.

[72] M. Schonlau and R. Y. Zou, "The random forest algorithm for statistical learning," *Stata Journal,* vol. 20, *(1),* 2020.

[73] T. Bailey and A. K. Jain, "Note On Distance-Weighted k-Nearest Neighbor Rules." *IEEE Transactions on Systems, Man and Cybernetics,* vol. SMC-8, *(4),* 1978. . DOI: 10.1109/tsmc.1978.4309958.

[74] T. M. Khoshgoftaar, S. Zhong and V. Joshi, "Enhancing software quality estimation using ensemble-classifier based noise filtering," *Intelligent Data Analysis,* vol. 9, *(1),* 2005.

[75] L. C. Briand *et al*, "Assessment and comparison of common software cost estimation modeling techniques," *Proceedings - International Conference on Software Engineering,* 1999.

[76] A. Idri, I. Abnane and A. Abran, "Evaluating Pred(p) and standardized accuracy criteria in software development effort estimation," *Journal of Software: Evolution and Process,* vol. 30, *(4),* 2018.

[77] M. Azzeh and A. B. Nassif, "A hybrid model for estimating software project effort from Use Case Points," *Applied Soft Computing Journal,* vol. 49, 2016.

[78] R. Silhavy, P. Silhavy and Z. Prokopova, "Evaluating subset selection methods for use case points estimation," *Information and Software Technology,* vol. 97, 2018.

[79] M. Azzeh and A. B. Nassif, "Analyzing the relationship between project productivity and environment factors in the use case points method," *Journal of Software: Evolution and Process,* vol. 29, *(9),* 2017.

[80] M. Azzeh and A. B. Nassif, "Project productivity evaluation in early software effort estimation," *Journal of Software: Evolution and Process,* vol. 30, *(12),* 2018.

[81] Sarwosri *et al*, "The development of method of the enhancement of technical factor (TF) and environmental factor (EF) to the use case point (UCP) to calculate the estimation of software's effort," in *Proceedings of 2016 International Conference on Information and Communication Technology and Systems, ICTS 2016,* 2017.

[82] M. Azzeh *et al*, "Ensemble of learning project productivity in software effort based on use case points," in *Proceedings - 17th IEEE International Conference on Machine Learning and Applications, ICMLA 2018,* 2019.

[83] M. Badri *et al*, "Source code size prediction using use case metrics: an empirical comparison with use case points," *Innovations in Systems and Software Engineering,* vol. 13, *(2-3),* 2017.

[84] Z. Prokopova, R. Silhavy and P. Silhavy, "The effects of clustering to software size estimation for the use case points methods," in *Advances in Intelligent Systems and Computing,* 2017.

[85] S. Bagheri and A. Shameli-Sendi, "Software project estimation using improved use case point," in *Proceedings - 2018 IEEE/ACIS 16th International Conference on Software Engineering Research, Management and Application, SERA 2018,* 2018.

[86] K. Qi and B. W. Boehm, "Detailed use case points (DUCPs): A size metric automatically countable from sequence and class diagrams," in *Proceedings - International Conference on Software Engineering,* 2018.

[87] H. T. Hoc, V. Van Hai and Le Thi Kim Nhung, Ho, "AdamOptimizer for the optimisation of use case points estimation," in *Advances in Intelligent Systems and Computing,* 2020.

[88] R. Silhavy, P. Silhavy and Z. Prokopova, "Improving algorithmic optimisation method by spectral clustering," in *Advances in Intelligent Systems and Computing,* 2017.

[89] A. B. Nassif *et al*, "Software development effort estimation using regression fuzzy models," *Computational Intelligence and Neuroscience,* vol. 2019, 2019.

[90] T. Vera, S. Ochoa and D. Perovich, "Survey of Software Development Effort Estimation Taxonomies," *Paper Knowledge Toward a Media History of Documents,* 2017.

[91] S. M. R. Chirra and H. Reza, "A Survey on Software Cost Estimation Techniques," *Journal of Software Engineering and Applications,* vol. 12, *(06),* 2019.

[92] A. J. Albrecht, "Measuring application development productivity," in *Joint SHARE/GUIDE/IBM Application Development Symposium,* 1979.

[93] A. J. Albrecht and J. E. Gaffney, "Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation," *IEEE Transactions on Software Engineering,* vol. SE-9, *(6),* 1983.

[94] L. H. Putnam, "A General Empirical Solution to the Macro Software Sizing and Estimating Problem," *IEEE Transactions on Software Engineering,* vol. SE-4, *(4),* 1978.

[95] A. Saeed *et al*, "Survey of software development effort estimation techniques," in *ACM International Conference Proceeding Series,* 2018.

[96] C. E. Carbonera, K. Farias and V. Bischoff, "Software development effort estimation: A systematic mapping study," *IET Software,* vol. 14, *(4),* 2020.

[97] P. Agrawal and S. Kumar, "Early phase software effort estimation model," in *Symposium on Colossal Data Analysis and Networking, CDAN 2016,* 2016.

[98] Available: www.ifpug.org, "Ifpug," in *IFPUG Couting Practices Manual,* 1986.

[99] C. Rush and R. Roy, "Expert judgement in cost estimating: Modelling the reasoning process," *Concurrent Engineering Research and Applications,* vol. 9, *(4),* 2001.

[100] M. Shepperd and C. Schofield, "Estimating software project effort using analogies," *IEEE Transactions on Software Engineering,* vol. 23, *(11),* 1997.

[101] M. Jørgensen, "Top-down and bottom-up expert estimation of software development effort," *Information and Software Technology,* vol. 46, *(1),* 2004.

[102] M. Cohn, "Agile estimating and planning," in *VTT Symposium (Valtion Teknillinen Tutkimuskeskus),* 2006.

[103] A. Ali and C. Gravino, "A systematic literature review of software effort prediction using machine learning methods," *Journal of Software: Evolution and Process,* vol. 31, *(10),* 2019.

[104] A. B. Nassif, L. F. Capretz and D. Ho, "Estimating software effort based on use case point model using sugeno fuzzy inference system," in *Proceedings - International Conference on Tools with Artificial Intelligence, ICTAI,* 2011.

[105] A. B. Nassif, L. F. Capretz and D. Ho, "Software effort estimation in the early stages of the software life cycle using a cascade correlation neural network model," in *Proceedings - 13th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, SNPD 2012,* 2012.

[106] M. S. Iraji and H. Motameni, "Object Oriented Software Effort Estimate with Adaptive Neuro Fuzzy use Case Size Point (ANFUSP)," *International Journal of Intelligent Systems and Applications,* vol. 4, *(6),* 2012.

[107] Ali Bou Nassif, Danny Ho, Luiz Fernando Capretz, "Regression model for software effort estimation based on the use case point method," *2011 International Conference on Computer and Software Modeling,* 2011.

[108] M. Ochodek, J. Nawrocki and K. Kwarciak, "Simplifying effort estimation based on Use Case Points," *Information and Software Technology,* vol. 53, *(3),* 2011.

[109] P. Pospieszny, B. Czarnacka-Chrobot and A. Kobylinski, "An effective approach for software project effort and duration estimation with machine learning algorithms," *Journal of Systems and Software,* vol. 137, 2018.

[110] Z. Abdelali, M. Hicham and N. Abdelwahed, "An ensemble of optimal trees for software development effort estimation," in *Lecture Notes in Networks and Systems*Anonymous 2019.

[111] Y. Mahmood *et al*, "Improving estimation accuracy prediction of software development effort: A proposed ensemble model," in *2nd International Conference on Electrical, Communication and Computer Engineering, ICECCE 2020,* 2020.

[112] A. Hussain *et al*, "Enhanced framework for ensemble effort estimation by using recursive-based classification," *IET Software,* vol. 15, *(3),* 2021.

[113] O. Malgonde and K. Chari, "An ensemble-based model for predicting agile software development effort," *Empirical Software Engineering,* vol. 24, *(2),* 2019.

[114] S. Shukla, S. Kumar and P. R. Bal, "Analyzing effect of ensemble models on multi-layer perceptron network for software effort estimation," in *Proceedings - 2019 IEEE World Congress on Services, SERVICES 2019,* 2019.

[115] I. Myrtveit and E. Stensrud, "Validity and reliability of evaluation procedures in comparative studies of effort prediction models," *Empirical Software Engineering,* vol. 17, *(1-2),* 2012.

[116] Le Thi Kim Nhung, Ho, H. T. Hoc and V. Van Hai, "An evaluation of technical and environmental complexity factors for improving use case points estimation," in *Advances in Intelligent Systems and Computing,* 2020.

[117] Nhung, Ho Le Thi Kim *et al*, "Parametric Software Effort Estimation Based on Optimizing Correction Factors and Multiple Linear Regression," *IEEE Access,* vol. 10, 2022.

[118] Nhung, Ho Le Thi Kim, V. Van Hai and R. Jašek, "Towards a correction factors-based software productivity using ensemble approach for early software development effort estimation," in *Computer Science on-Line Conference,* 2022.

[119] Tibshirari R., "Regression Shrinkage and Selection via the Lasso," *Journal of the Royal Statistical Society, Series B,* vol. 58, *(1),* 2013.

[120] P. Bühlmann and S. v. d. Geer, *Statistics for High-Dimensional Data: Methods, Theory and Applications.* 2011.

[121] B. Anda *et al*, "Estimating software development effort based on use cases – experiences from industry," in *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics),* 2001.

[122] A. Chandra and X. Yao, "Ensemble learning using multi-objective evolutionary algorithms," *Journal of Mathematical Modelling and Algorithms,* vol. 5, *(4),* 2006.

[123] T. G. Dietterich, "Ensemble methods in machine learning," in *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics),* 2000.

[124] Z. Ma and Q. Dai, "Selected an Stacking ELMs for Time Series Prediction," *Neural Processing Letters,* vol. 44, *(3),* 2016.

[125] X. Luo *et al*, "Short-term wind speed forecasting via stacked extreme learning machine with generalized correntropy," *IEEE Transactions on Industrial Informatics,* vol. 14, *(11),* 2018.

[126] A. I. Naimi and L. B. Balzer, "Stacked generalization: an introduction to super learning," *European Journal of Epidemiology,* vol. 33, *(5),* 2018.

[127] D. R. Jeffery and M. J. Lawrence, "Some issues in the measurement and control of programming productivity," *Information and Management,* vol. 4, *(4),* 1981.

[128] T. Urbanek, A. Kolcavova and A. Kuncar, "Inferring productivity factor for use case point method," in *Annals of DAAAM and Proceedings of the International DAAAM Symposium,* 2017.

[129] W. J. Schneider. G, "Applied use cases," in *A Practical Guide, Second Edition, Addison-Wesley,* 2001.

[130] K. An and J. Meng, "Voting-averaged combination method for regressor ensemble," in *International Conference on Intelligent Computing,* 2010.

[131] Van der laan, M J and S. Dudoit, "Unified cross-validation methodology for selection among estimators and a general cross-validated adaptive epsilon-net estimator: Finite sample oracle inequalities and examples," *U.C. Berkeley Division of Biostatistics Working Paper,* vol. Working Pa, 2003.

[132] Vaart, Aad W. van der, S. Dudoit and Laan, Mark J. van der, "Oracle inequalities for multi-fold cross validation," *Statistics & Decisions,* vol. 24, *(3),* 2006.

[133] B. Anda, E. Angelvik and K. Ribu, "Improving estimation practices by applying use case models," in *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics),* 2002.

[134] C. L. Huang and C. J. Wang, "A GA-based feature selection and parameters optimizationfor support vector machines," *Expert Systems with Applications,* vol. 31, *(2),* 2006.

[135] I. H. Witten *et al*, *Data Mining: Practical Machine Learning Tools and Techniques.* 2016.

LIST OF PUBLICATIONS OF THE AUTHOR

Journal papers:

1. H.LT.K. Nhung, V.V. Hai, R. Silhavy, Z. Prokopova, and P. Silhavy, "Parametric Software Effort Estimation Based on Optimizing Correction Factors and Multiple Linear Regression, " IEEE Access, vol. 10, pp. 2963-2986, DOI: 10.1109/ACCESS.2021.3139183, 2022.
2. V.V. Hai, H.L.T.K. Nhung, Z. Prokopova, R. Silhavy, and P. Silhavy, "A New Approach to Calibrating Functional Complexity Weight in Software Development Effort Estimation," Computers 11, no. 2: 15, DOI: 10.3390/computers11020015, 2022.
3. V.V. Hai, H.L.T.K. Nhung, Z. Prokopova, R. Silhavy, and P. Silhavy, "Towards improving the efficiency of software development effort estimation via clustering analysis," IEEE Access, vol. 10, pp. 83249-83264, DOI: 10.1109/ACCESS.2022.3185393, 2022.

Conference papers:

4. H.LT.K. Nhung, V.V. Hai, and R. Jasek, "Towards a Correction Factors-based Software Productivity using Ensemble approach for Early Software Development Effort Estimation," Lecture Notes in Networks ans Systems, vol. 501 LNNS, pp. 413-425, DOI: 10.1007/978-3-031-09070-7_35, 2022.
5. H.LT.K. Nhung, V.V. Hai, and H.T. Hoc, "Analyzing Correlation of the relationship between Technical Complexity Factors and Environmental Complexity Factors for Software Development Effort Estimation", Lecture Notes in Networks and Systems, 232 LNNS, pp. 835-848, DOI: 10.1007/978-3-030-90318-3_65, 2021.
6. H.LT.K. Nhung, V.V. Hai, and H.T. Hoc, "Evaluation of Technical and Environmental Complexity Factors for Improving Use Case Points Estimation," Advances in Intelligent Systems and Computing Springer, 1294, pp. 757–768, DOI: 10.1007/978-3-030-63322-6_64, 2020.
7. H.L.T.K. Nhung, H.T. Hoc, and V.V. Hai, "A Review of Use Case-Based Development Effort Estimation Methods in the System Development Context," Advances in Intelligent Systems and Computing, 1046, pp. 484-499, DOI: 10.1007/978-3-030-30329-7_44, 2019.
8. V.V. Hai, H.L.T.K. Nhung, and R. Jasek, "Toward appying agglomerative hierarchical clustering in improving the software development effort estimation," Lecture Notes in Networks and Systems, vol. 501 LNNS, pp. 353-371, DOI: 10.1007/978-3-031-09070-7_30, 2022.
9. V.V. Hai, H.L.T.K. Nhung, Z. Prokopova, R. Silhavy, and P. Silhavy, “Analyzing the effectiveness of the Gaussian Mixture Model clustering algorithm in Software Enhancement Effort Estimation,” ACIIDS 2022.

10. V.V. Hai, H.L.T.K. Nhung, H.T. Hoc, "Calibrating Function Complexity in Enhancement Project for Improving Function Points Analysis Estimation," Lecture Notes in Networks and Systems, 232 LNNS, pp. 857-869, DOI: 10.1007/978-3-030-90318-3_67, 2021.
11. V.V. Hai, H.L.T.K. Nhung, H.T. Hoc, "Empirical Evidence in Early Stage Software Effort Estimation Using Data Flow Diagram," Lecture Notes in Networks and Systems, 230, pp. 632-644, DOI: 10.1007/978-3-030-77442-4_53, 2021.
12. V.V. Hai, H.L.T.K. Nhung, H.T. Hoc, "A Productivity Optimising Model for Improving Software Effort Estimation," Advances in Intelligent Systems and Computing, 1294, pp. 735-746, DOI: 10.1007/978-3-030-63322-6_62, 2020.
13. V.V. Hai, H.L.T.K. Nhung, H.T. Hoc, "A Review of Software Effort Estimation by Using Functional Points Analysis," Advances in Intelligent Systems and Computing, 1047, pp. 408-422, DOI: 10.1007/978-3-030-31362-3_40, 2019.
14. H.T. Hoc, V.V. Hai, H.L.T.K. Nhung, "An Approach to Adjust Effort Estimation of Function Point Analysis," Lecture Notes in Networks and Systems, 230, pp. 522-537, DOI: 10.1007/978-3-030-77442-4_45, 2021.
15. H.T. Hoc, V.V. Hai, H.L.T.K. Nhung, "AdamOptimizer for the Optimisation of Use Case Points Estimation," Advances in Intelligent Systems and Computing, 1294, pp. 747-756, 2020.
16. H.T. Hoc, V.V. Hai, H.L.T.K. Nhung, " A Review of the Regression Models Applicable to Software Project Effort Estimation," Advances in Intelligent Systems and Computing, 1047, pp. 399-407, 10.1007/978-3-030-31362-3_39, 2019.

CURRICULUM VITAE AUTHOR

Name:	Ho Le Thi Kim Nhung
Education and degrees:	Tomas Bata University in Zlin, Czech Republic PhD student, Software Engineering 12/2018-Now
	University of Science, Vietnam (HCMUS-VNU) Master of Information Systems 2011-2014
	University of Science, Vietnam (HCMUS-VNU) Bachelor of Information Systems 2007-2010
Related Work Experience:	University of Science, Vietnam (HCMUS-VNU) Lecturer, Faculty of Information Technology 2011-2018
	Institute of International Management (IIMBA), National Cheng Kung University, Tainan, Taiwan Visiting research scientist 8/2016-8/2017
	National Institute of Informatics, Japan Visiting research scientist 8/2016-1/2017
Honors and Awards:	Outstanding Young Lecturer at University of Science HCMUS-VNU (2015, 2017)
	Top 10 graduate of Honors degree at University of Science (HCMUS-VNU), Certificate of Merit on being the best student of graduation (7/500 students) (2010)