

A Systematic Analysis of Data Preprocessing for Machine Learning- based Software Cost Estimation

Jianglin Huang, Yan-Fu Li, Min Xie

▶ **To cite this version:**

Jianglin Huang, Yan-Fu Li, Min Xie. A Systematic Analysis of Data Preprocessing for Machine Learning- based Software Cost Estimation. Information and Software Technology, 2015, 67, pp.108-127. 10.1016/j.infsof.2015.07.004 . hal-01340341

HAL Id: hal-01340341
https://hal.science/hal-01340341

Submitted on 30 Jun 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

A Systematic Analysis of Data Preprocessing for Machine Learning-based Software Cost Estimation

Jianglin Huang*, Yan-Fu Li, Min Xie

- *J. Huang and M. Xie are with the Department of Systems Engineering and Engineering Management, City University of Hong Kong, Tat Chee Avenue, Hong Kong SAR, China*
  E-mails: jianhuang7-c@my.cityu.edu.hk, minxie@cityu.edu.hk
- *Y. F. Li is with the Laboratory of Industrial Engineering, Ecole Centrale Paris, Grande Voie des Vignes 92295, Chatenay-Malabry Cedex, France.*
  E-mail: yanfu.li@ecp.fr

**corresponding author. Tel: +85256013641*

Abstract

Context: Due to the complex nature of the software development process, traditional parametric models and statistical methods often appear to be inadequate to model the increasingly complicated relationship between project development cost and the project features (or cost drivers). Machine learning (ML) methods, with several reported successful applications, have gained popularity for software cost estimation in recent years. Data preprocessing has been claimed by many researchers as a fundamental stage of ML methods; however, very few works have been focused on the effects of data preprocessing techniques.

Objective: This study aims for a systematic assessment of the effectiveness of data preprocessing techniques on ML methods in the context of software cost estimation.

Method: In this work, we first conduct a literature survey of the recent publications using data preprocessing techniques, followed by a systematic empirical study to analyze the strengths and weaknesses of individual data preprocessing techniques as well as their combinations.

Results: Our results indicate that data preprocessing techniques may significantly influence the final prediction. They sometimes might have negative impacts on prediction performance of ML methods.

Conclusion: In order to reduce prediction errors and improve efficiency, a careful selection is necessary according to the characteristics of machine learning methods, as well as the datasets used for software cost estimation.

Keywords – software cost estimation, data preprocessing, missing-data treatments, scaling, feature selection, case selection

1. Introduction

Software project managers often need to estimate the cost/effort of developing a software system at the early stage of its life-cycle [1] in order to plan the project management activities. The ability to accurately estimate the development cost plays an important role in the success of software project management. In the past decades, numerous research works have been published on software cost estimation (SCE) methods, which can be classified into the following three main categories.

1. *Expert judgment*: It requires the consultation of one or more experts to derive the cost estimate [2]. With the experience and available information of past projects and the understanding of a new project, the experts could obtain the estimation by a non-explicit and subjective reasoning process. It is the most frequently applied method for software projects in practice [3].
2. *Parametric models*: They often involve the utilization of analytical or statistical equations relating software project cost to a number of project features. The well-known ones include COCOMO [4] and SLIM Model [5].
3. *Machine Learning* (*ML*) *methods*: They involve at least one modeling method, taking a number of project features and producing a cost prediction, making no or minimal assumptions about the form of the relation under study. Thus they can provide higher approximation capabilities to solve complex problems. Recently, they have been adopted as an alternative or together with the first two methods [6-10]. Representative ML methods include artificial neural networks (ANN) [8, 11, 12], case-based reasoning (CBR) [12, 13] (also referred to as analogy-based estimation [14, 15] or estimation by analogy [16]), and classification and regression trees (CART) [7, 17, 18].

When targeting estimation accuracy, considerable effort has been devoted to improving ML methods [1, 19-24]. For the empirical validations, ML algorithms are routinely tested on the SCE datasets. Data preprocessing (DP) is a fundamental stage of the ML application, which has been reported to have significant impacts onto the

performances of ML methods [21].

To the knowledge of the authors, there is very few research work focused on the DP techniques in the SCE literature. In many situations the DP techniques, such as feature selection (FS) [10, 25-27] and case selection (CS) [6, 14, 15, 28], have been considered as a necessary step for CBR while for other ML methods, such as ANN and CART, they might be ignored. In the literature some studies focus on analyzing DP techniques. Strike et al. [29] simulated various incomplete data and found that the best regression model could be obtained from missing-data imputation with Z-score scaling. The combination of scaling scheme and missing-data treatment (MDT) is firstly analyzed; however, their impacts onto the ML method were not studied. Many studies propose one or more DP techniques to deal with a specific issue in SEC, such as data missingness [30-32], redundant or irrelevant features [13, 28], or abnormal cases [33, 34]. But they did not study the effectiveness of different DP techniques. Keung et al. [35] first time concluded that the performance of a ML method could be significantly altered by a DP technique, such as scaling and FS. But the number of DP techniques they considered is limited and the effectiveness of combined DP techniques are not investigated.

From the analysis above, a systematic study on multiple DP techniques for ML methods is needed to promote much more careful use of the DP techniques rather than taking one or more DP approaches as granted. The empirical results obtained would be beneficial to the following research works who adopt ML methods for SCE.

The rest of this paper is organized as follows: Section 2 presents a literature survey on DP applications; Section 3 presents the four datasets used in this study, an overview on the ML methods (*i.e.* ANN, CBR and CART), and the experimental design; Section 4 presents the experiment results and analysis; Section 5 discusses the threats to four types of validity; Section 6 concludes this work and points out future research directions.

2. Related Work

2.1 *Literature survey*

The application of ML algorithm requires the presence of data in a mathematically feasible format through data preprocessing. DP techniques consist of data reduction, data projection and missing-data treatment. Data reduction aims to decrease the size of the datasets by means of feature selection (FS) or case selection (CS). Data projection intends to transform the appearance of the data, *e.g.* scaling, which scales all features into a pre-defined same range. Missing-data treatments (MDTs) include deleting missing values [15, 18, 19, 36, 37] and/or replacing them with the estimates [16, 38, 39]. Moreover, the logarithm transformation [40] is also regarded as one data projection method and is frequently applied for linear regression due to the normality assumption of the linear model. Logarithm is mostly used by regression studies to ensure the normality of the residual. For ML methods, it is not a commonly applied DP technique. In our survey, there are only three publications [35, 41-43] that used logarithm for ML methods. Considering its relatively infrequent utilization by the researchers, we choose to not include it as a candidate DP method in our experiments which aim to investigate the effectiveness of the popular DP techniques for ML methods.

To reveal the situations of DP technique utilization in the literature, we first conduct a survey of relevant ML papers from 2005 to present published on the following journals: *IEEE Transactions on Software Engineering* (*IEEE TSE*), *Empirical Software Engineering* (*ESE*), *Journal of Systems Software* (*JSS*), *Information and Software Technology* (*IST*), and *Software Quality Journal* (*SQJ*), and the following major conference proceedings: *International Conference on Software Engineering* (*ICSE*), *International Symposium on Empirical Software Engineering and Measurement* (*ESEM*), and *International Conference on Predictive Models in Software Engineering* (*PROMISE*). Both the individual studies of ML methods and the comparative studies (within ML methods or between ML and other methods) are

included. We summarize the publications according to the ML methods and the DP techniques applied. In specific, we explore the use of MDT, scaling, FS and CS. These 48 publications are presented in Table 1. It is shown that most publications have employed certain DP techniques. 12 works [6, 11, 14, 43-51] only mention single step of DP. Table 1 also shows that many studies use combined DPs. For examples, there are 7 of totally 48 works combined only scaling and FS/CS [20, 27, 34, 52-55], and 7 of 48 works combined only MDTs and FS/CS [19, 23, 36, 56-59]. FS and CS have been considered as a necessary step for CBR in several studies [6, 14, 23, 26, 33, 37, 47, 48, 52, 55-57, 60-63]. However, there is no systematic study to investigate the DPs and their combinations.

Table 1

Data preprocessing techniques used by ML methods for SCE

Source	Reference	Methods	FS[a]/CS[b]	Scaling[c]	MDT[d]
IEEE TSE	Pendharkar *et al.* (2005) [49]	ANN, BBN, CART	FS		
	Auer *et al*. (2006) [50]	CBR		[0, 1]	
	Keung *et al.* (2008) [26]	CBR	FS, CS	[0, 1]	LD
	Kocaguneli *et al.* (2012) [14]	CBR	FS, CS		
	Kocaguneli *et al.* (2012) [41]	CART, SVM	FS	[0, 1]	MI
	Kocaguneli *et al.* (2013) [64]	CBR, CART	FS	[0, 1]	MI
	Menzies *et al.* (2013) [51]	CBR	FS		
	Mittas and Angelis (2013) [65]	ANN, CART, CBR	FS		LD
JSS	Chiu and Huang (2007) [17]	ANN, CART, CBR		[0, 1]	LD
	de Barcelos Tronto *et al.* (2008) [11]	ANN		[0, 1]	
	Vinary *et al.* (2008) [27]	ANN	FS	[0, 1]	
	Li *et al.* (2009) [28]	ANN, CART, CBR, SVM	CS	[0, 1]	LD
	Azzeh *et al.* (2011) [15]	CBR		[0, 1]	LD
	Bou *et al.* (2013) [6]	ANN	FS, CS		
IST	Huang and Chiu (2006) [18]	ANN, CART, CBR	FS	[0, 1]	LD
	Mittas *et al.* (2008) [44]	ANN		[0, 1]	
	Oliveira *et al.* (2010) [20]	SVM, GP	FS	[0, 1]	
	Minku and Yao (2013) [61]	ANN, CART, CBR	FS, CS	[0, 1]	MI
ESE	Li *et al*. (2007) [38]	CBR	FS	[0, 1]	LD
	Li and Ruhe (2008) [19]	CBR	FS		LD
	Azzeh *et al.* (2009) [36]	CBR, ANN	FS		LD
	Li *et al.* (2009) [54]	ANN	FS	[0, 1]	
	Mittas and Angelis (2010) [37]	CBR	FS, CS	[0, 1]	LD
	Lopez-Martin *et al.* (2011) [66]	ANN	CS		
	Azzeh (2012) [62]	CBR	FS, CS	[0, 1]	LD
	Corazza *et al.* (2013) [59]	CBR, SVM	FS		LD
	Kocaguneli *et al.* (2013) [16]	CBR	FS	[0, 1]	MI
	Seo and Bae (2013) [33]	CBR	FS, CS	[-1, 1]	LD
SQJ	Liu *et al.* (2008) [57]	CART, CBR	FS, CS		LD
	Bakır *et al.* (2009) [67]	SVM, CBR	FS	[0, 1]	LD
	Hsu and Huang (2010) [63]	ANN, CART, CBR	FS, CS	[0, 1]	LD
	Bakır *et al.* (2011) [68]	CART, CBR	FS	[0, 1]	LD
	Khatibi et al. (2013) [58]	CBR	FS		LD
ICSE	Ramasubbu and Balan (2012) [69]	CBR	CS		

ESEM	Li *et al.* (2007) [45]	CBR			MI
	Mendes (2007) [46]	BBN, CBR, CART	FS		
	Keung (2008) [47]	CBR	FS, CS		
	Corazza *et al.* (2009) [43]	SVM, CBR, BBN	CS		
	Kocaguneli and Menzies (2011) [53]	CBR	CS	[0, 1]	
PROMISE	Li and Ruhe (2007) [60]	CBR	FS, CS		MI
	Brady and Menzies (2010) [52]	CBR	FS, CS	[0, 1]	
	Corazza *et al.* (2010) [42]	ANN, SVM, CBR, BBN	CS	[0, 1]	
	Minku and Yao [23]	ANN, CART	FS, CS		*k*-NN
	Tsunoda *et al.* (2011) [34]	CBR	CS	[0, 1]	LD
	Borges and Menzies (2012) [55]	CBR, RI	FS, CS	[0, 1]	
	Ferrucci *et al.* (2012) [48]	CBR	FS, CS		
	Kocaguneli *et al.* (2012) [56]	CART, CBR	FS, CS		MI
	Kosti *et al.* (2012) [70]	CBR	CS	[0, 1]	LD

ANN: *Artificial Neural Networks*, BBN: *Bayesian Belief Network*, CART: *Classification and Regression Trees*, CBR: *Case-Based Reasoning*, GP: *Genetic Programming*, RI: *Rule Induction*, SVM: *Support Vector Machine*

[a]FS/ [b]CS: if Feature Selection or/and Case Selection are used;

[c]Scaling: [0, 1], [-1, 1], or in blank (Not specific);

[d]MDT: which Missing-Data Treatment is used, LD (Listwise Deletion), MI (Mean Imputation), *k*-NN imputation or in blank (Not specific).

The following section presents an overview on the four types of DP methods and summarizes the evidence and arguments in the literature that lead to the research questions of this study and serve as the foundation for the empirical work.

2.2 Data preprocessing techniques

2.2.1 Missing-data treatments

Due to the high cost of gathering and reporting data from projects, development teams are less focused on data collection [31]. The incomplete datasets also frequently appear across the SCE datasets (*e.g.* the ISBSG database and PROMISE datasets) [16, 24, 31, 32, 37, 58, 71]. The missing values have significant impacts on ML estimation performances, as reported by [22, 31, 45].

There are many MDTs in the literature. They often include: deletion methods (*listwise deletion* and *pairwise deletion* [31, 49, 70, 71]), and imputation methods (*mean imputation*, *k-NN imputation*, *hot-deck imputation*, *cold-deck imputation*, *regression imputation* and *multiple imputation methods*) [16, 23, 38, 55, 56, 60]. It is noted that the deletion methods, especially listwise deletion (LD), widely used as a default approach dealing with missing-values, can result in discarding large

proportions of datasets in cases and introducing biasness [31, 37]. As another solution of MDT, imputation requires more extensive and complicated statistical and computational analysis [29, 31] and also includes natural prediction error [37]. Mean imputation (MI) imputes each missing value with the mean of observed values and preserves the information of data. However, as the simplest imputation method it may cause to diminish the variance of variables [29].

According to results of our survey in Table 1, LD is the most popular method followed by MI. Particularly, several works [15, 18, 19, 24, 34, 36, 37, 57-59, 63, 65, 67] regarded LD as the default DP method for missing values. However, some studies show that MI or k-NN imputations are better than LD [29-31, 72-74]. In this study, we will validate the superiority of MI over LD.

2.2.2 Scaling

Scaling generally refers to measurements or assessments conducted under exact, specified and repeatable conditions. In ML, scaling transforms feature values according to a defined rule so that all scaled features have the same degree of influence [40] and thus the method is immune to the choice of units [70], which is a major stage for ML methods [75]. Normally, the intervals of [0, 1] and [-1, 1] are used to be the target of scaling, as shown in *Eq.* (1).

$$\begin{aligned} [0,1]\ \text{interval} &= \frac{actualValue - \min(allValues)}{\max(allValues) - \min(allValues)} \\ [-1,1]\ \text{interval} &= \frac{actualValue - (\max(allValues) + \min(allValues))/2}{(\max(allValues) - \min(allValues))/2} \end{aligned} \tag{1}$$

Z-score standardization is also used for scaling [29], which makes the unit of the variables the sample standard deviation. Scaling is essential since every sub-sequential process is dependent upon the choice of unit for each feature. For example, it will be improper to let *Lines of Code* (*LOC*) be a thousand times more influential than *KLOC*. In SCE context, scaling is the most popular preprocessing technique.

As shown in Table 1, most of the studies that consider scaling use [0, 1] and only

one study uses [-1, 1]. The survey also revealed that [0, 1] scheme is used by most researchers by default without questioning whether it is truly effective or not, and many studies do not even consider or mention any type of scaling during the estimation processes. For example, in [76] the scaling scheme of [0, 1] was conducted without reasoning.

2.2.3 Feature selection

FS, also known as attribute selection or feature subset selection, is the process of selecting a subset of features that have significant or similar impacts onto the evaluation target as using all features [63, 77]. It assumes the data contains irrelevant or/and redundant features [78] that might decrease the model performance. FS was initially proposed to increase the accuracy of induced classifier in supervised learning algorithm [78]. Later, Mendes *et al.* [79] and Chen *et al.* [77] introduced FS to SCE. An exhaustive search for all possible subsets of features is calculated by $2^N - 1$, in which N is the total number of project features. Take ISBSG dataset [80] as example, since $n=15$ (See in Section 3.1.1), there are 32767 possible subsets. Obviously, exhaustively exploring all the subsets will not be cost-effective.

More intelligent FS techniques have been developed. They can be classified as wrappers and filters. The former often convolves with predictors, using cross-validation to predict the benefits of adding or removing a feature from the feature subset used [78]. In SCE literature, wrapper methods utilize random searching, hill climbing, forward sequential selection [13, 26] and reduce research method [19] to guide the search process. As the predictors (*i.e.* ML methods) are often expensive to run, the wrappers are generally time-consuming and possibly leads to over-fitting [26, 78]. The latter selects features by evaluating some preset criteria independently prior to training the ML methods, which has much lower time complexity than the former does. For example, variable ranking with the correlation coefficient is a filter method: it is independent of the choice of the predictor [81]. However, the selected features often yield considerable prediction errors [82].

In SCE literature, there are a number of studies that propose new FS techniques or investigate the effectiveness of the existing ones. In general, majority of the researchers [13, 19, 20, 26, 36, 77, 83] support the use of FS with ML methods. In this study, we will check the validity of applying FS technique. Our survey also reveals that FSS is the most popular FS algorithm, due to its simplicity and the ease of adaptation to various ML methods. There are 19 of 48 publications (shown in Table 1) using this strategy [14, 19, 36-38, 41, 46-48, 54-57, 60, 62, 64-66, 68].

2.2.4 Case selection

Similar to the features of a dataset, not all the historical cases are useful for estimating the present project. CS aims to identify and remove redundant and irrelevant cases [84]. By reducing the entire dataset into a smaller subset that consist only of representative cases, this method could save computing time and produce prediction results comparable to those using all the cases. Searching techniques for FS are also applicable to CS, *e.g.* random search, hill climbing, simulated annealing, forward and backward sequential selection [60, 84], genetic algorithm [28], and filters [81, 82]. On contrary to the selection of useful cases, the abnormal cases detection [33, 34] is also regarded as one type of CS. Compared to FS, the CS treatment is relatively infrequent but with a rising trend in the recent literature. In the first CS study, Kirsopp and Shepperd [84] presented a conservative backward sequential selection (BSS) and proved its effectiveness. In this study we will implement this technique since discarding a large number of projects at the initiate stage of FSS could result in loss of useful information.

2.3 *Research questions*

Based upon the above bibliographic analysis the following 5 research questions (RQs) are raised for the empirical study:

RQ1: Is MDT effective? Which commonly used MDT is preferred?

RQ2: Is scaling effective? [0, 1] scheme or [-1, 1], which is preferred?

RQ3: Is FS effective? Shall we consider it for all ML methods?

RQ4: Is CS effective? Shall we consider it as well?

RQ5: Is any combination of DP techniques effective? How do we select the DP techniques for each ML method and each dataset?

The above questions are answered in Section 4: Experimental results.

3. Experiment Design

In this section, we describe the datasets, error metrics and experiment procedures for our empirical studies.

3.1 Dataset description

In this study, we have selected the ISBSG database, Desharnais dataset, and Kitchenham dataset from PROMISE datasets for empirical tests.

3.1.1 ISBSG dataset

ISBSG has developed and refined its data collection standard over 10 years based on the metrics that have proven to be most useful in helping to improve software development management and processes. In this study, we adopted the ISBSG R10 data repository [80], which contains totally 6000 projects (with 105 features) coming from 24 countries and various organizations. Due to the heterogeneous nature and large size of the entire repository, ISBSG recommends extracting a suitable subset for any SCE practice [80]. In this study, we follow ISBSG's suggestion to select a subset with 14 features including 7 numerical project size features, and the following categorical features: '*development type*', '*primary programming language*', '*development platform*', '*organization type*', '*business area type*', '*application type*', and '*development techniques*' [80]. As suggested by [71, 85] that data in ISBSG needs to be appropriately prepared for further usage, the following steps are taken to remove the cases with minor significance to our analysis:

1. Select the project with A or B rating in *data quality* as well as A rating in *UFP* as suggested by ISBSG [80] and related published papers [37, 57, 65, 86, 87] . The rest with lower rating are excluded.
2. Filter out the projects with normalized ratio larger than 1.2, as it is suggested by ISBSG that the ratio up to 1.2 is more acceptable [80]. The normalized ratio defined as *normalized effort* divided by *summary effort*, is constructed by ISBSG for the refinement of project subset.
3. Select the project with resource level equal to '1'. As shown in ISBSG dataset [80], a resource level of '1' means that only the effort of development team is recorded; the levels above add peripheral efforts. Similar to related studies [33, 66, 71, 88], this study is only interested in the work effort of development team.
4. Select the project using IFPUG as the functional sizing method. The functional sizing method is used to compute the number of *AFP* (*Adjusted Function Points*), an essential feature in most SCE study. The projects in ISBSG repository sized using 4 types of functional sizing methods: IFPUG, NESMA, MARK II, COSMIC-FFP, among which IFPUG gains the highest popularity [37, 61, 63, 66]. Furthermore, ISBSG suggests users do not mix the projects sized using pre-IFPUG with the ones sized using IFPUG V4/V4+. Therefore, we only keep the projects sized using IFPUG V4/V4+.

After these four procedures, we obtain a subset of 446 projects with a considerable amount of missing values. Among all features, '*NorEffort*' is to be estimated. The descriptive statistics of the numerical features are summarized in Table 2. It shows that the ISBSG subset is of high order non-normality. Note that in Table 2, *w/o* indicates that the statistics in the column are obtained by excluding the missing values.

Table 2

Descriptive statistics of numeric features of ISBSG dataset

Features	Mean		Std Dev		Min		Max		Skewness		Kurtosis	
	*w/**	*w/o**	*w/*	*w/o*	*w/*	*w/o*	*w/*	*w/o*	*w/*	*w/o*	*w/*	*w/o*
DevType	1.4	1.8	0.5	0.5	1	1	3	3	1.0	-0.3	-0.1	0.3
OrgType	4.7	5.9	4.6	4.0	-1	1	17	16	1.0	1.0	0.2	-0.1
BusType	0.3	3.0	2.0	1.4	-1	1	7	7	1.4	1.4	1.2	1.2
AppType	5.0	5.3	5.1	5.6	-1	1	19	19	1.0	1.4	0.0	0.8
DevPlat	0.9	1.6	1.3	1.0	-1	1	4	4	0.2	1.7	0.2	1.7

PriProLan	4.8	7.7	4.9	6.6	-1	1	19	19	1.2	0.4	0.4	-1.5
DevTech	0.1	2.3	1.8	1.8	-1	1	7	6	1.8	1.0	3.2	-0.6
InpCont	58.7	243.7	452.8	1082.1	-1	0	9404	9404	19.9	8.4	410.4	72.1
OutCont	27.2	112.4	83.2	169.0	-1	0	1221	1221	8.3	4.4	101.5	25.3
EnqCont	23.3	71.8	67.6	116.9	-1	0	893	893	6.6	5.0	67.3	33.1
FileCont	28.1	120.7	149.4	345.0	-1	0	2955	2955	17.2	7.7	333.2	63.7
IntCont	4.5	16.9	17.2	27.5	-1	0	160	135	5.1	2.4	31.1	6.0
AFP	348.9	627.0	980.3	2014.9	6	43	17518	17518	13.5	8.2	220.0	69.3
NorEffort	5165.6	3531.9	10140.5	6647.1	91	300	134211	54620	7.1	6.5	70.4	48.4

**w/: with missing values (missing values are denoted as '-1'); *w/o: without missing values*

3.1.2 Desharnais dataset

The Desharnais dataset [12, 89] has been used in many research works [12, 25, 41], due to its scalability and availability. It includes 81 projects (4 with missing values) and 11 features. The features include '*TeamExp*', '*ManagerExp*', '*YearEnd*', '*Length*', '*Transactions*', '*Entities*', '*PointsAdjust*', '*Envergure*', '*PointsNonAjust*', '*Language*' and '*Effort*', among which the dependent value '*Effort*' will be estimated. Table 3 shows the statistics of the numerical features in details.

Table 3

Descriptive statistics of numeric features of Desharnais dataset

Features	Mean		Std Dev		Min		Max		Skewness		Kurtosis	
	*w/**	*w/o**	*w/*	*w/o*	*w/*	*w/o*	*w/*	*w/o*	*w/*	*w/o*	*w/*	*w/o*
TeamExp	2.2	2.3	1.4	1.3	-1	0	4	4	-0.2	-0.1	-1.0	-1.3
ManagerExp	2.5	2.6	1.6	1.5	-1	0	7	7	0.0	0.2	0.1	0.1
YearEnd	85.8	85.8	1.1	1.1	83	83	88	88	-0.2	-0.2	0.0	0.1
Length	11.7	11.3	7.4	6.8	1	1	39	36	1.6	1.5	3.1	2.7
Language	1.6	1.6	0.7	0.7	1	1	3	3	0.9	0.9	-0.5	-0.5
Transactions	179.9	177.5	143.3	146.1	9	9	886	886	2.4	2.4	7.7	7.7
Entities	122.3	120.5	84.9	86.1	7	7	387	387	1.3	1.4	1.5	1.5
PointsAdjust	302.2	298.0	179.7	182.3	73	73	1127	1127	1.8	1.8	4.9	5.1
Envergure	27.6	27.5	10.6	10.5	5	5	52	52	-0.1	-0.2	-0.3	-0.4
PointsNonAdjust	287.0	282.4	185.1	186.4	62	62	1116	1116	1.7	1.7	4.2	4.4
Effort	5046.3	4833.9	4418.8	4188.2	546	546	23940	23940	2.0	2.0	4.7	5.3

**w/: with missing values (missing values are denoted in as '-1'); *w/o: without missing values*

3.1.3 Kitchenham dataset

The Kitchenham dataset includes 145 projects of a software development company. There are only three features to be recommended as independent variables, which are *Project type*, *Actual duration*, and *AFP*, plus the dependent attribute *Actual effort* [34]. Table 4 shows the statistics of the numerical features in details.

Table 4

Descriptive statistics of numeric features of Kitchenham dataset

Features	Mean		Std Dev		Min		Max		Skewness		Kurtosis	
	*w/**	*w/o**	*w/*	*w/o*	*w/*	*w/o*	*w/*	*w/o*	*w/*	*w/o*	*w/*	*w/o*

Project type	1.4	1.5	0.9	0.7	-1	1	4	-0.3	1.5	5.0	5.5
Actual duration	527.7	527.8	1522	1572.9	15.4	18.9	181400	10.8	10.5	125.3	118.1
AFP	206.7	201	134.1	130.6	37	37	950	1.9	2.1	8.9	10.5
Actual effort	3113.1	3169.1	9598	9333.6	219	219	113930	10.8	10.4	124.3	116.2

**w/: with missing values (missing values are denoted in as '-1'); *w/o: without missing values*

3.1.4 USPFT dataset

The USPFT (University Student Projects developed in 2005) dataset includes 76 projects, which were collected from student projects on web and client/server applications. As referred from Li, Ruhe, Al-Emran and Richter's work in [38], the projects with a feature (FT, a set of requirements) as objective type are used as USPFT. Table 5 shows the statistics of the numerical features in details.

Table 5

Descriptive statistics of numeric features of USPFT dataset

Features	Mean		Std Dev		Min		Max		Skewness		Kurtosis	
	*w/**	*w/o**	*w/*	*w/o*	*w/*	*w/o*	*w/*	*w/o*	*w/*	*w/o*	*w/*	*w/o*
Internal Process	19.9	19.9	17.5	17.5	0	0	100		1.6	1.6	7.6	7.6
Data E/M/D	26.2	26.2	13.8	13.8	0	0	80		0.9	0.9	7.5	7.5
Output form	21.5	21.5	17.2	17.2	0	0	100		2.5	2.5	12.4	12.4
Data Query	22.5	22.5	11.6	11.6	0	0	54		-0.1	-0.1	3.0	3.0
Printing	1.6	1.6	5.0	5.0	0	0	30		3.5	3.5	17.0	17.0
Report	2.3	2.3	4.8	4.8	0	0	20		2.2	2.2	7.3	7.3
Other	5.6	5.6	13.3	13.3	0	0	100		4.9	4.9	34.2	34.2
IntComplx	1.9	1.9	1.2	1.2	1	1	5		1.1	1.1	3.1	3.1
DataFile	3.3	3.3	3.3	3.3	0	0	18		2.0	2.0	7.4	7.4
DataEn	17.4	17.4	48.3	48.3	0	0	314		4.6	4.5	25.4	25.4
DataOut	1.9	2.0	6.1	6.2	-1	0	50		6.6	6.6	50.3	49.3
UFP	11.7	12.1	26.0	26.3	-1	0	180		4.0	3.9	24.2	23.6
Lang	2.6	2.7	0.8	0.6	-1	1	4		-2.1	-1.2	8.6	4.5
Tools	2.7	2.8	0.8	0.5	-1	1	4		-2.7	-1.4	12.5	6.8
ToolExpr	10.7	11.0	11.6	11.6	-1	0	52.5		1.8	1.8	5.4	5.3
AppExpr	2.5	2.5	1.4	1.4	1	1	5		0.4	0.4	1.7	1.7
TeamSize	1.7	1.7	0.7	0.7	1	1	4		1.4	1.3	4.8	4.8
DBMS	1.5	1.8	1.1	0.5	-1	1	3		-1.3	0.0	3.9	2.7
Method	2.2	2.9	1.7	0.8	-1	1	4		-1.0	-0.9	2.5	3.6
AppType	2.6	2.8	1.1	0.6	-1	1	4		-2.3	-1.3	8.2	5.7
Effort	5.5	5.5	8.7	8.7	0.5	0.5	40		2.4	2.4	8.5	8.5

**w/: with missing values (missing values are denoted in as '-1'); *w/o: without missing values*

3.2 *Machine learning methods in study*

In SCE literature, ML methods receive increasing attention in the recent years according to a comprehensive review done by Jorgensen and Shepperd [90]. Additionally, Section 2.1 shows that ANN, CBR, and CART are mostly applied in recent studies. We therefore select them for experiments.

Case-based reasoning CBR has been extensively studied and implemented due to its

simplicity and effectiveness [9, 17, 18, 40, 50, 79, 91]. Generally, a CBR consists of the following three steps:

1. Collect the measurements of past projects and prepare the historical projects dataset.
2. Select the relevant features, and then calculate the distances of the selected features between the project x' being estimated and the *i*-th historical projects x_i, and retrieve *K* nearest neighbors. The Euclidean distance, $D(x_i, x') = \sqrt{\sum_{j=1}^{p} Dis(x_{ij}, x'_j)}$, and Manhattan distance, $D(x_i, x') = \sum_{j=1}^{p} Dis(x_{ij}, x'_j)$, are considered in this study. $Dis(x_{ij}, x'_j)$ is defined as follows,

$$Dis(x_{ij}, x'_j) = \begin{cases} (x_{ij} - x'_j)^2, & \text{for Euclidean distance, if feature } x_{ij} \text{ and } x'_j \text{ are numeric or ordinal} \\ |x_{ij} - x'_j|, & \text{for Manhattan distance, if feature } x_{ij} \text{ and } x'_j \text{ are numeric or ordinal} \\ 1, & \text{if feature } x_{ij} \text{ and } x'_j \text{ are nominal and } x_{ij} = x'_j \\ 0, & \text{if feature } x_{ij} \text{ and } x'_j \text{ are nominal and } x_{ij} \neq x'_j \end{cases} \quad (2)$$

3. Compute the final prediction for the new project x' based on the selected *K* nearest neighbors. We use mean, median and inverse distance weighted mean (IDWM) as adaptation techniques in this study. The mean is the classical measure of central tendency and treats all analogies as being equally influential on the cost estimates. The median is more robust statistic when the number of neighbors increases [40]. IDWM [10] allows more close neighbors to have more influence than less close ones.

Artificial neural network Due to the capability of good approximation, ANN has become a frequently applied methodology in the context of SCE [7, 8, 92]. A typical feed-forward ANN architecture consists of a number of neurons connected through the input layer, hidden layer(s), and output layer. A typical three layer ANN has the following mathematical form

$$y = \hat{f}(x) = f(\sum_{j=1}^{J} w_j f_j(\sum_{i=1}^{p} v_{ij} x_i + \alpha_j) + \beta) \quad (3)$$

where $x \in R^p$, $f(\cdot)$ is the transfer function, J is the number of hidden nodes, v_{ij} is the connection weight between the i-th input node and j-th hidden node, α_j is the bias of the j-th hidden node, w_j is the weight on the link between the j-th hidden node and the output neuron, and β is the bias. The classical back-propagation algorithm is commonly used [66] to update the weights and biases with the attempt to minimize the predictive error [20, 92]. The user-defined parameters in ANN, *i.e.* the number of hidden layers, the number of hidden nodes a**n**d the type of transfer function, have significant impacts on ANN prediction performance. In this study, only one hidden layer is used since multiple layers may lead to an over parameterized ANN structure which is often prone to over-fitting.

Classification and regression trees CART is capable of dealing with both numerical and categorical features to identify major subsets [7]. The construction of CART involves recursively splitting the data set into relatively homogeneous subsets until the terminate conditions are satisfied. The algorithm starts with all the training instances in the root node and then selects an independent feature A_i that best divides the training set into disjoint subsets. The partition is determined by minimizing the Mean Square Error (MSE) of the dependent feature (*e.g.*, project cost). Suppose the feature f_i partition the entire training data set X into subsets X_{ij} where each sample takes the same value for feature f_i. The MSE of any subset X_{ij} with the dependent value y_k is

$$MSE\left(X_{ij}\right) = \sum_{k \in T_{ij}} (y_k - \bar{y})^2 / \left|X_{ij}\right| \tag{4}$$

where $\bar{y}$ is the mean of the y_k values exhibited in X_{ij} and $\left|X_{ij}\right|$ is the size of the set X_{ij}. The feature that minimizes $\sum_i MSE\left(X_{ij}\right)$ is selected for partition. This process is repeated until each node reaches a user-specified minimum tree size. The primary control parameter in CART, level of tree pruning, is optimized by the cross-validation scheme.

3.3 *Error metrics*

Error evaluation criteria are essential to the evaluation of prediction. Three popular error metrics: *Mean of Balanced Relative Error* (MBRE), PRED(0.25), and *Median of Balanced Relative Error* (MdBRE) are also used in this study. Their building block: *Balanced Relative Error* (BRE) of the estimated *i*-th project is defined as

$$BRE_i = \left| \frac{y_i - \hat{y}_i}{\min(y_i, \hat{y}_i)} \right| \quad (5).$$

Based on *Eq.* (5), MBRE is defined as

$$MBRE = \sum_{i=1}^{n} BRE_i / n \quad (6)$$

and MdBRE is defined as

$$MdBRE = median(BRE_i) \quad (7)$$

MBRE is a balanced symmetric error metric [93]. MdBRE, exhibiting a pattern similar to that of MBRE but less sensitive to extreme outliers, is more likely to select the true model especially in the underestimation cases [94, 95]. PRED(0.25) is the percentage of predictions that fall within 25 percent of the actual cost. It is defined as

$$PRED(0.25) = \frac{1}{n} \sum_{i=1}^{n} \begin{cases} 1, if\ BRE_i \le 0.25 \\ 0, \quad otherwise \end{cases} \quad (8)$$

PRED(0.25) identifies the SCE methods that are generally accurate.

MBRE has been widely used for accuracy measuring due to its advantages: applicability to all kinds of prediction models across all datasets, independence to measurement units and scales [29, 91]. Compared with MBRE and MdBRE, PRED exhibits an opposite pattern. These three evaluation criteria are widely used in recent SCE studies [37, 54, 61, 74, 96, 97]. In our experiments, we consider MBRE as the guidance for the parameter optimization of the ML methods on validating datasets. The analysis in Section 3.5 shows that it is significantly correlated with both PRED(0.25) and MdBRE, which confirms that MBRE could be a representative error

metric for parameter optimization.

3.4 Experiment setting

For each testing dataset, we generate 24 processed versions by using different combinations of DP techniques. The FS, CS, scaling and MDT introduced in Section 2.2 are regarded as four main factors in our experiment design. First, for missing values we include the MDTs discussed in the review: listwise deletion (LD) and mean imputation (MI). Then, for scaling we employ two scaling schemes, [0, 1] and [-1, 1], to evaluate possible effects of scaling into different intervals. Finally, as Kirsopp and Shepperd [84] suggested (see in Section 2.2.4), we adopt forward sequential selection (FSS) search strategy for FS and backward sequential selection (BSS) search strategy for CS. Under each of the main factors except MDTs, we setup the alternative 'Null' as control, which means no corresponding preprocessing is performed. Consequently, 24 processed datasets are generated by this design. Table 6 presents the details of the design.

Table 6
Generation of processed datasets

Preprocessing techniques		**FS**	Null		FSS	
		CS	Null	BSS	Null	BSS
MDT	**Scaling**					
LD	Null		#1	#7	#13	#19
	[0, 1]		#2	#8	#14	#20
	[-1, 1]		#3	#9	#15	#21
MI	Null		#4	#10	#16	#22
	[0, 1]		#5	#11	#17	#23
	[-1, 1]		#6	#12	#18	#24

MDT: missing-data treatment; LD: listwise deletion; MI: mean imputation; FS: feature selection; CS: case selection; FSS: forward sequential selection; BSS: backward sequential selection; Null: no DP is used.

It should be noted that the processing techniques are applied on only the project features (independent variables). The cost values (dependent variable) are scaled into the range [0, 1] regardless of the type of processing technique applied on features.

3.5 *Cross-validation*

As discussed in the overview of ML methods, the predefined parameters have large impacts on their performances. A systematic way of determining the predefined parameters should be considered in the experiments. A simple but effective way for parameter tuning is the so-called cross-validation scheme. In this scheme, the entire dataset is randomly split into three equally sized and mutually exclusive subsets: training subset, validating subset and testing subset. The training subset is used to construct models with specified parameter settings. The validating subset is used for parameter tuning and to prevent over-fitting problem of ML methods. The testing subset is used to evaluate the prediction abilities of training methods with optimal parameters. The above validation scheme is performed three times on each processed dataset to eliminate the biases from different split, similarly to [79].

Table 7

Predefined parameters in meta-modeling techniques

Methods	Predefined parameters	Range
CBR	1. Distance functions	{Euclidean distance, Manhattan distance}
	2. Number of nearest neighbors	$\{k \mid k = 1, 2, 3, 4, 5\}$
	3. Adaptation functions	{Mean, Median, Inverse distance weighted mean}
ANN	1. Number of hidden neurons	$\{n \mid n = 1, 3, 5, 7, 9, 11, 13\}$
	2. Hidden layer transfer function	{Linear, Tan-Sigmoid, Log-Sigmoid}
CART	Level of tree pruning	$\{l \mid l = 1, 2, 3, 4, 5\}$

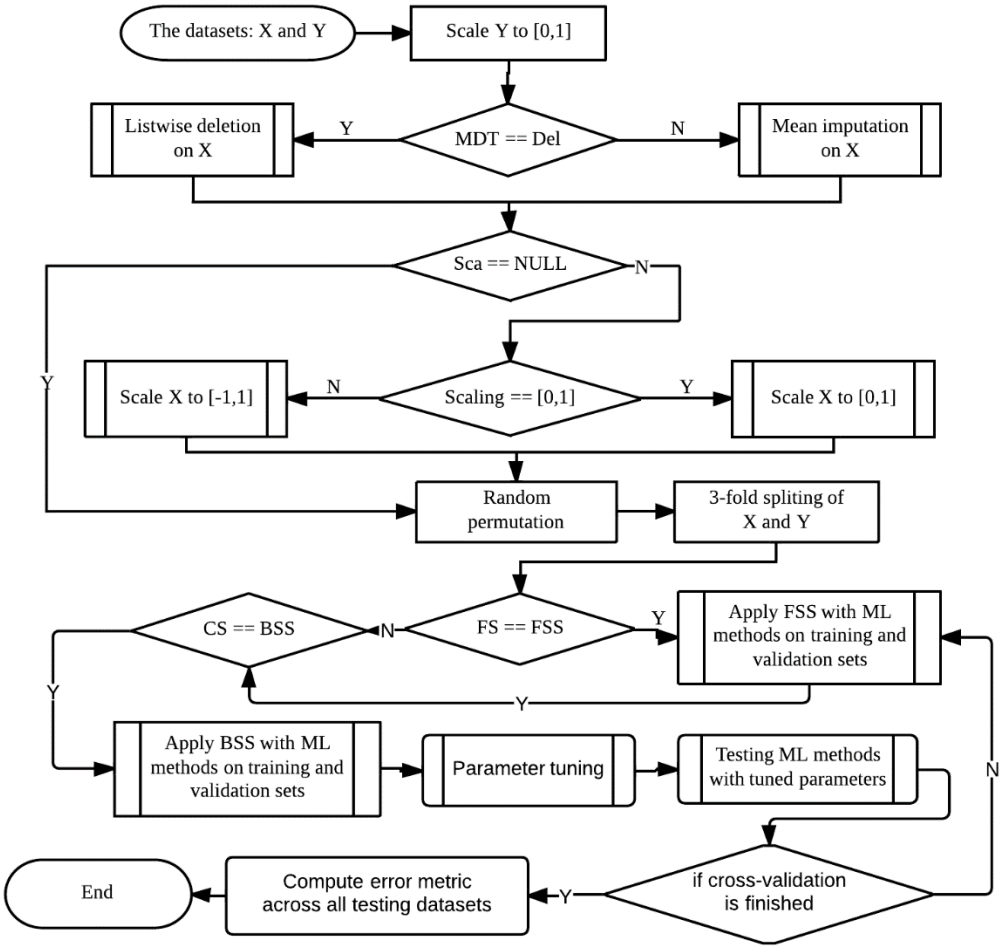


Fig. 1 Experiment process

Table 7 summarizes the predefined parameters of each method and their ranges for tuning. The CBR parameters ranges cover most existing ones appeared in CBR studies. For ANN, the maximal number of hidden neurons is set to 13 because too many hidden neurons may lead to an over-parameterized ANN structure. The three types of hidden layer transfer functions are most frequently used in ANN models. ANN is trained by the back-propagation algorithm. The training stops when the MSE drops below the specified threshold (0.0001 in this study). For CART, the level of tree pruning is in the range from 1 to 5, to balance modeling accuracy and over-fitting. The training stops when its MSE is below 0.0001. Note that all the methods in this study are implemented via MATLAB programming. Fig. 1 illustrates the overall experiment procedures. After MDT and scaling steps, the cross-validation scheme starts. At each fold, FS is applied with ML methods on the training and the validation

sets followed by CS treatment, then the parameters are tuned on validation set and finally the methods are tested on the testing set. After 3 times of repetitions, the testing results across all data splits are aggregated to compute MBRE, PRED and MdBRE.

4. Experiment Results

Experiment results are first analyzed by investigating the marginal impact of each individual preprocessing technique on the ML performances. As discussed in section 2.2, the techniques considered in this study are: FS (Null/FSS), CS (Null/BSS), Scaling (Null/[0, 1]/[-1, 1]), and MDT (LD/MI). First, the means and standard deviations of the error metrics (MBRE, PRED(0.25) and MdBRE) across the testing subsets (i.e. three random splits of each dataset) by each technique are collected and analyzed. The mean values of error metrics indicate one method's average accuracy on a group of processed datasets. The standard deviations of error metrics reflect the robustness of one method across different datasets. The 'PC' column in tables refers to the percentage of change between comparisons. Then, ANOVA is conducted to examine the significance of each preprocessing technique and their combinations. Finally, the overall performances of ML methods are compared and analyzed.

4.1 Performance under MDT scheme

The marginal means and standard deviations of the testing errors under two MDT schemes are shown in

Table 8. For ISBSG dataset with relatively large number of missing values, MI could enlarge MBRE of all ML methods. The testing error of ANN has some outliers under MI, as the mean MBREs are not consistent with mean PREDs and MdBREs. For Desharnais dataset with a small amount of missing values, only ANN has slightly decreased MBRE value by MI. Furthermore, MI generally performs worse than LD in terms of accuracy and robustness of CBR and CART under MBRE. However, the testing errors of CBR and CART have outliers under MI as well. For Kitchenham

dataset with also a small amount of missing values, only CBR has increased MBRE by MI. The testing errors in terms of PRED and MdBRE are consistent with the result of MBRE. MI could generally enhance the robustness of ANN and CART on Kitchenham dataset. On USPFT dataset with a medium number of missing values, the results are similar to that on ISBSG data. The testing error of CART has outliers under MI in terms of the results of PREDs and MdBREs are not consistent with the MBREs.

The results indicate that the effectiveness of MI compared with LD depend on the datasets as well as the ML methods. CBR is consistently negatively influenced by MI. CART is sensitive to outliers, and ANN is less sensitive to small amount of missing values. The answers to RQ1 from the results are the following: 1) it might not be correct to apply the same LD or MI to all ML methods on all datasets (as most of the previous studies did); 2) the MDT needs to be carefully selected for each ML method on each dataset. To the knowledge of the authors, none of the existing studies have made such discoveries.

Table 8

Average accuracy and robustness under MDT scheme

Testing Error Metrics	ML Methods	Mean			Std.			Mean			Std.		
		LD	MI	*PC (%)	LD	MI	*PC (%)	LD	MI	*PC (%)	LD	MI	*PC (%)
Dataset		ISBSG						Kitchenham					
MBRE	CBR	0.365	6.79	+1762	0.0623	4.13	+6535	2.39	2.62	+9.70	0.306	0.344	+12.5
	ANN	0.531	9.39	+1668	0.749	2.36	+215	12.9	12.7	-1.55	10.3	7.82	-24.5
	CART	0.349	5.96	+1606	0.0242	0.912	+3658	2.72	2.50	-8.27	0.441	0.262	-40.6
PRED	CBR	0.487	0.477	-2.05	0.0831	0.0390	-53.0	0.243	0.218	-10.6	0.0461	0.0552	+19.8
	ANN	0.544	0.563	+3.47	0.0746	0.0302	-59.4	0.198	0.206	+4.03	0.0435	0.0426	-2.31
	CART	0.520	0.497	-4.27	0.0440	0.0376	-14.5	0.212	0.241	+13.3	0.0181	0.0110	-39.2
MdBRE	CBR	0.269	0.274	+1.82	0.0598	0.0333	-44.2	0.736	0.847	+15.1	0.113	0.245	+116
	ANN	0.231	0.215	-6.84	0.0419	0.0171	-59.3	0.947	0.912	-3.70	0.287	0.252	-11.8
	CART	0.237	0.254	+7.25	0.0364	0.0304	-16.4	0.807	0.771	-4.58	0.0618	0.0502	-18.8
Dataset		Desharnais						USPFT					
MBRE	CBR	11.7	16.3	+39.0	1.28	12.1	+847	138	222	+60.5	74.8	124	+65.9
	ANN	51.2	51.0	-0.391	20.1	17.5	-12.9	330	352	+6.81	165	194	+17.3
	CART	23.9	27.8	+16.0	14.9	20.4	+36.5	140	160	+14.1	48.5	53.2	+9.79
PRED	CBR	0.467	0.475	+1.66	0.0418	0.0558	+33.5	0.656	0.564	-13.9	0.0671	0.0907	+35.0
	ANN	0.507	0.513	+1.16	0.0333	0.0465	+39.6	0.422	0.354	-16.2	0.0981	0.102	+4.28
	CART	0.461	0.506	+9.78	0.0311	0.0353	+13.6	0.431	0.500	+16.0	0.167	0.0651	-60.8
MdBRE	CBR	0.282	0.266	-5.63	0.0451	0.0395	-12.1	0	0.135	+∞	0	0.138	+∞
	ANN	0.252	0.249	-1.23	0.0271	0.0323	+19.3	0.431	0.528	+22.4	0.167	0.254	+52.6
	CART	0.281	0.252	-10.3	0.0153	0.0162	+5.94	0.328	0.282	-13.8	0.0843	0.0555	-34.1

**PC (%) is the percentage of change between LD and MI.*

4.2 Performance under Scaling scheme

The marginal means and marginal standard deviations of the testing errors under different scaling schemes are shown in Table 9. The results exhibit diverse patterns in each dataset. On all datasets, the scaling scheme has no impact on CART. The result of ISBSG dataset shows the testing errors are not consistent with MBREs although scaling schemes generally improve the accuracy and robustness of MBREs. On Desharnais dataset, the scaling schemes reduce testing error and improve robustness of ANN method. CBR, instead, is impacted by outliers. On Kitchenham dataset with a small number of features, CBR is uniquely negatively impacted by scaling schemes. On USPFT dataset, the [0, 1] scheme improves the accuracy and robustness of CBR and ANN. The [0, 1] scheme outperforms [-1, 1] scheme consistently in terms of ANN.

Table 9

Average accuracy and robustness under scaling scheme

Dataset	Testing Error Metrics	ML Methods	Mean					Std.				
			Null	[0, 1]	*PC (%)	[-1, 1]	*PC (%)	Null	[0, 1]	*PC (%)	[-1, 1]	*PC (%)
ISBSG	MBRE	CBR	4.47	3.14	-29.8	3.13	-30.0	6.05	3.51	-42.1	3.47	-42.6
		ANN	5.26	4.51	-14.1	5.11	-2.61	4.73	4.54	-4.12	5.79	+22.4
		CART	3.15	3.15	0	3.15	0	3.07	3.07	0	3.07	0
	PRED	CBR	0.495	0.479	-3.13	0.473	-4.38	0.0837	0.0508	-39.2	0.0586	-29.8
		ANN	0.558	0.560	+0.358	0.548	-1.77	0.0676	0.0629	-6.97	0.0437	-35.2
		CART	0.508	0.508	0	0.508	0	0.0435	0.0435	0	0.0435	0
	MdBRE	CBR	0.269	0.271	+0.631	0.274	+1.92	0.0594	0.0432	-27.3	0.0443	-25.3
		ANN	0.221	0.219	-0.905	0.226	+2.69	0.0365	0.0358	-1.92	0.0253	-30.6
		CART	0.246	0.246	0	0.246	0	0.0355	0.0355	0	0.0355	0
Desharnais	MBRE	CBR	18.8	11.4	-39.2	11.8	-37.0	14.5	1.15	-92.1	0.948	-93.4
		ANN	56.9	42.6	-25.1	53.6	-5.75	24.8	14.3	-42.2	13.1	-47.3
		CART	25.8	25.8	0	25.8	0	18.4	18.4	0	18.4	0
	PRED	CBR	0.500	0.462	-7.56	0.451	-9.62	0.0564	0.0301	-46.5	0.0462	-18.1
		ANN	0.491	0.531	+7.85	0.507	+3.18	0.0479	0.0302	-36.9	0.0333	-30.4
		CART	0.483	0.483	0	0.483	0	0.0417	0.0417	0	0.0417	0
	MdBRE	CBR	0.243	0.286	+17.9	0.294	+21.0	0.0363	0.0306	-15.7	0.0431	+18.5
		ANN	0.265	0.234	-11.7	0.253	-4.48	0.0317	0.0226	-28.8	0.0267	-15.7
		CART	0.267	0.267	0	0.267	0	0.0225	0.0225	0	0.0224	0
Kitchenham	MBRE	CBR	2.22	2.65	+19.5	2.66	+19.5	0.0866	0.339	+292	0.338	+290
		ANN	15.6	13.0	-17.1	8.89	-43.2	9.02	10.3	+14.5	7.11	-21.1
		CART	2.59	2.59	0	2.59	0	0.374	0.374	0	0.374	0
	PRED	CBR	0.265	0.213	-19.5	0.214	-19.2	0.0319	0.0541	+69.6	0.0527	+65.1
		ANN	0.191	0.219	+14.7	0.196	+2.53	0.0635	0.0222	-65.1	0.0289	-54.4
		CART	0.227	0.227	0	0.227	0	0.0223	0.0223	0	0.0223	0
	MdBRE	CBR	0.643	0.867	+34.8	0.866	+34.7	0.0289	0.208	+621	0.208	+619
		ANN	1.13	0.788	-30.3	0.868	-23.3	0.351	0.132	-62.2	0.124	-64.7
		CART	0.782	0.782	0	0.782	0	0.0640	0.0640	0	0.0640	0
USPFT	MBRE	CBR	171	140	-18.1	229	+33.7	124	122	-1.80	61.7	-50.4
		ANN	321	289	-9.85	414	+29.0	169	99.8	-41.1	233	+37.8

	CART	150	150	0	150	0	53.2	53.2	0	53.2	0
PRED	CBR	0.619	0.626	+1.12	0.585	-5.57	0.0765	0.0668	-12.7	0.125	+64.3
	ANN	0.385	0.440	+14.2	0.339	-11.7	0.104	0.0765	-26.9	0.114	+8.82
	CART	0.465	0.465	0	0.465	0	0.134	0.134	0	0.134	0
MdBRE	CBR	0.0643	0.0337	-43.7	0.104	+210	0.0906	0.0551	-39.1	0.178	+222
	ANN	0.436	0.389	-10.9	0.614	+40.5	0.149	0.136	-8.83	0.284	+89.8
	CART	0.305	0.305	0	0.305	0	0.0770	0.0770	0	0.0770	0

**PC (%) is the percentage change between the [0, 1] and NULL or the [-1, 1] and NULL.*

The results reveal that [0, 1] scheme of appears to be less effective to highly-skewed ISBSG and Kitchenham datasets (described in Section 3.1) than to the less-skewed Desharnais and USPFT datasets. The influences of both [0, 1] and [-1, 1] schemes for CART are ignorable for all datasets. This might be attributed to the fact that the scaling does not change the distributions of the original data and consequently does not impact the splitting results of CART. Additionally, the [0, 1] scheme is more suitable than [-1, 1] scheme for ANN in all datasets. For CBR, only the results obtained from USPFT dataset are improved by [0, 1] scheme consistently. The discussions in this paragraph provide answers to RQ2.

4.3 *Performance under FS scheme*

Table 10 illustrates the marginal means and standard deviations of the three error metrics on testing subsets classified by FS scheme: Null/FSS. The results show that FSS could generally reduce the testing error and improve robustness for CBR method. In addition, FS is more efficient on Desharnais dataset in terms of reducing testing error and enhancing robustness of CBR and ANN methods. For ISBSG dataset, ANN and CART appear to reduce prediction accuracy and robustness in terms of MBRE. However, for CART method on all datasets, FS generally has a negative impact. For Kitchenham dataset with relative small number of features, FSS also shows negative impact on ANN method. On USPFT dataset, the prediction accuracy and robustness of CBR and ANN in terms of MBRE have been improved; however, the testing errors are not consistent.

This analysis shows that FS generally improve CBR method (especially for ISBSG, Desharnais and Kitchenham data) and it should be used with care for ANN.

Applying FS for CART might not be a good choice. For datasets with small number of features, applying FS for ANN could overall reduce the number of hidden nodes and therefore hinder its approximation ability. Nevertheless, these might attribute to the limitation of specific FS approach used in this study. The discussions in this paragraph provide answers to RQ3.

Table 10

Average accuracy and robustness under FS scheme

Testing Error Metrics	ML Methods	Mean			Std.			Mean			Std.		
		Null	FSS	*PC (%)	Null	FSS	*PC (%)	Null	FSS	*PC (%)	Null	FSS	*PC (%)
Dataset		ISBSG						Kitchenham					
MBRE	CBR	5.11	2.05	-59.7	5.52	2.03	-63.1	2.67	2.35	-11.8	0.362	0.236	-34.8
	ANN	4.79	5.13	+7.16	4.36	5.46	+25.2	11.8	13.2	+11.7	7.47	10.5	+41.7
	CART	2.78	3.53	+27.0	2.53	3.36	+33.1	2.55	2.67	+4.40	0.409	0.341	-16.4
PRED	CBR	0.467	0.498	+6.87	0.0649	0.0608	-6.32	0.211	0.250	+18.4	0.0536	0.0431	-19.4
	ANN	0.531	0.577	+8.60	0.0578	0.0469	-18.7	0.190	0.215	+12.9	0.0391	0.0434	+11.1
	CART	0.506	0.511	+0.812	0.0297	0.0522	+75.7	0.219	0.233	+6.71	0.0185	0.0206	+11.2
MdBRE	CBR	0.282	0.261	-7.51	0.0473	0.0470	-0.759	0.879	0.705	-19.7	0.238	0.0843	-64.5
	ANN	0.237	0.209	-11.6	0.0346	0.0239	-31.1	1.02	0.839	-17.7	0.237	0.269	+13.3
	CART	0.247	0.243	-1.88	0.0261	0.0414	+59.2	0.820	0.757	-7.64	0.0512	0.0482	-6.02
Dataset		Desharnais						USPFT					
MBRE	CBR	14.8	13.2	-10.4	11.7	4.61	-60.7	242	118	-51.1	94.8	86.5	-8.75
	ANN	60.8	41.3	-32.1	18.0	13.2	-26.3	367	315	-13.9	198	156	-21.2
	CART	23.4	28.3	+20.6	14.4	20.6	+42.1	136	164	+20.8	51.8	47.7	-7.77
PRED	CBR	0.454	0.488	+7.48	0.0498	0.0427	-14.4	0.599	0.621	+3.67	0.0389	0.124	+220
	ANN	0.490	0.529	+8.12	0.0401	0.0288	-28.0	0.413	0.363	-12.1	0.0666	0.129	+95.1
	CART	0.487	0.480	-1.39	0.0240	0.0522	+116	0.500	0.431	-13.9	0.175	0.0340	-80.5
MdBRE	CBR	0.278	0.271	-2.16	0.0475	0.0380	-19.9	0.0388	0.0965	+148	0.0632	0.152	+141
	ANN	0.266	0.236	-11.2	0.0275	0.0232	-15.6	0.418	0.541	+29.5	0.109	0.278	+153
	CART	0.267	0.265	-0.758	0.0179	0.0252	+40.5	0.303	0.307	+1.52	0.0966	0.0443	-54.1

**PC (%) is the percentage change between FSS and NULL.*

4.4 Performance under CS scheme

Table 11 presents the performance of the ML methods under CS scheme: Null/BSS. On ISBSG dataset, CS appears to lead to diverse results: for ANN, the testing error is reduced and robustness is improved for all metrics; for CBR and CART, the situation is the opposite to that of ANN. On Desharnais dataset, CS could increase testing error and reduce robustness of ANN, and reduce the testing error of CART. As to CBR on Desharnais dataset, the result shows the testing error is not consistent with the MBRE. On Kitchenham dataset, CS appears to lead to negative impact on CART and positive impact on ANN. For CBR, the MBRE is slightly

reduced but the testing errors are not consistent. On USPFT dataset, only CART is consistently improved by CS. For ANN and CBR, MBREs are not consistent with the testing errors. It is possibly due to the outliers inside the dataset.

Table 11

Average accuracy and robustness under CS scheme

Testing Error Metrics	ML Methods	Mean			Std			Mean			Std		
		Null	BSS	*PC (%)	Null	BSS	*PC (%)	Null	BSS	*PC (%)	Null	BSS	*PC (%)
Dataset		ISBSG						Kitchenham					
MBRE	CBR	2.89	4.26	+47.5	3.36	5.23	+55.6	2.52	2.50	-0.817	0.413	0.266	-35.4
	ANN	5.31	4.60	-13.3	5.31	4.52	-14.8	12.6	12.4	-1.82	8.71	9.65	+10.7
	CART	3.09	3.22	+4.04	2.87	3.12	+8.61	2.34	2.88	+23.1	0.332	0.144	-56.5
PRED	CBR	0.501	0.463	-7.58	0.0465	0.0742	+59.5	0.252	0.209	-17.1	0.0322	0.0592	+84.0
	ANN	0.551	0.556	+0.915	0.0579	0.0576	-0.431	0.185	0.218	+17.7	0.0398	0.0396	-0.502
	CART	0.510	0.507	-0.658	0.0338	0.0497	+47.1	0.231	0.221	-4.18	0.0215	0.0193	-10.2
MdBRE	CBR	0.255	0.288	+12.7	0.0341	0.0544	+59.3	0.738	0.845	+14.4	0.121	0.243	+101
	ANN	0.224	0.222	-0.571	0.0349	0.0310	-11.1	1.01	0.849	-16. 1	0.306	0.196	-36.1
	CART	0.244	0.246	+0.641	0.0300	0.0388	+29.3	0.776	0.801	+3.14	0.0468	0.0677	+44.5
Dataset		Desharnais						USPFT					
MBRE	CBR	16.8	11.2	-32.8	11.9	1.30	-89.1	164	196	+19.4	100	119	+18.3
	ANN	49.2	53.1	+7.96	13.5	22.8	+68.0	309	373	+20.6	181	174	-3.70
	CART	42.16	9.601	-77.2	8.28	0.0716	-99.1	160	140	-12.1	53.4	48.4	-9.44
PRED	CBR	0.477	0.465	-2.33	0.0532	0.0447	-15.9	0.604	0.616	+2.05	0.115	0.0634	-44.8
	ANN	0.505	0.515	+2.05	0.0341	0.0456	+34.0	0.384	0.392	+2.16	0.119	0.0917	-23.1
	CART	0.455	0.512	+12.5	0.0299	0.0258	-13.4	0.406	0.524	+29.2	0.131	0.0984	-24.9
MdBRE	CBR	0.273	0.275	+0.826	0.0411	0.0452	+9.89	0.101	0.0344	-65.8	0.146	0.0722	-50.7
	ANN	0.253	0.248	-1.99	0.0247	0.0341	+37.7	0.481	0.478	-0.765	0.251	0.186	-25.6
	CART	0.281	0.252	-10.3	0.0167	0.0155	-7.22	0.346	0.263	-23.9	0.0841	0.0218	-74.1

**PC (%) is the percentage change between BSS and NULL.*

The results show different patterns across the datasets in the study. CBR unlikely benefit from CS. CART may benefit from CS on within-company datasets, like Desharnais and USPFT datasets, whereas CS might have negative impacts to CART on cross-company datasets, such as ISBSG and Kitchenham datasets. CS could be used by ANN for cross-company datasets. However, it is not always effective when attached to CART or ANN, possibly due to the loss of information for training. The discussions in this paragraph provide answers to RQ4.

4.5 ANOVA test

To further quantify the impact and significance of each preprocessing technique as well as the interactions between them, we conduct Analysis of Variance (ANOVA) for this purpose. MBRE is used as the sole response (or dependable variable) for ANOVA.

We analyze Spearman's rho non-parametric correlations between the individual error metrics (MBRE, PRED(0.25) and MdBRE) across all testing datasets. The analysis reveals a significant positive correlation at the level of 0.01 between MBRE and MdBRE, and a negative correlation at the level of 0.01 between MBRE and PRED(0.25). Consequently, the use of MBRE as response solely appears to be feasible. On top of the significance testing, the non-linear interactions between the main effects are also investigated. The factors of ANOVA including "Method" (CBR, ANN or CART), MDT (LD/MI), Scaling (Null/[0, 1]/[-1, 1]), FS (Null/FSS) and CS (Null/BSS), are all transformed to dummy variable and regarded as fixed factors. The interactions in this study include ten 2-ways, ten 3-ways, and five 4-ways. The factors/interactions are regarded as influential only if they have the significance level of 0.05 at least. Prior to the ANOVA, the logarithmic transformation is performed in response to reducing the skewness and the heterogeneity of the variances of different groups of MBRE values.

The ANOVA results are presented separately in Table 12, which lists all the main effects and significant interactions. The main effects and their interactions to explain a proportion of the total variance is measured by the partial eta squared statistic (η_p^2) with larger values related to higher relative importance. For ISBSG dataset, the main effects of 'Method', FS and MDT are proven to be significant at the level of 0.01. Moreover, Scaling is only significant at the level of 0.05. It confirms our finding from previous individual analysis that MDT, scaling, forward sequential feature selection appear to have strong influence on testing performances on ISBSG dataset. With respect to the interactions, two 2-way interactions: 'MDT × CS', 'Scaling × FS', three 3-way interactions: 'MDT × Scaling × CS', 'Method × Scaling × FS', and 'Method × MDT × FS', and two 4-way interaction: 'Method × MDT × FS × CS' and 'Method × Scaling × FS × CS' are significant at the level of 0.05.

For Desharnais dataset, only the main effects of 'Method' and CS are shown to be significant at the level of 0.01. With respect to the interactions, a 2-way interactions

and a 3-way interaction are significant at the level of 0.05, which are 'MDT × CS' and 'MDT × Scaling × CS', respectively. Furthermore, on Desharnais dataset, MDT appears to have strong interactions with CS.

For Kitchenham dataset, all main effects except the FS and CS are shown to be significant at the level of 0.05. It also confirms the findings that the preprocessing methods of FS and CS have less consistent influence on the prediction results. For USPFT dataset, all the main effects are shown to be significant at the level of 0.05 at least. Many interactions are shown to be significant at the level of 0.05.

In general, the analysis recognizes the single DP methods of MDT, FS and scaling significantly could impact the final prediction results for the most datasets or ML methods.

Table 12

ANOVA results of main effects and relevant interactions

Dataset	Factors/ Interactions	*d.f.*	Type III Sum *Sq.*	Mean *Sq.*	*F*	*p*-value	η_p^2
ISBSG	Method	2	1.028	.514	27.510	.005**	.932
	MDT	1	155.101	155.101	8303.128	.000**	.999
	Scaling	2	.238	.119	6.365	.050*	.761
	FS	1	1.228	1.228	66.740	.001**	.943
	CS	1	.007	.007	.374	.574	.086
	MDT× CS	1	.208	.208	11.160	.029*	.736
	Scaling × FS	2	.423	.211	11.320	.023*	.850
	MDT × Scaling × CS	2	.336	.168	8.992	.033*	.818
	Method × Scaling × FS	4	.462	.115	6.182	.050*	.861
	Method × MDT × FS	2	1.651	.825	44.190	.002**	.957
	Method × MDT × FS × CS	2	.518	.259	13.864	.016*	.874
	Method × Scaling × FS × CS	4	.500	.125	6.694	.046*	.870
Desharnais	Method	2	21.409	10.701	208.600	.000**	.991
	MDT	1	.144	.144	2.814	.169	.413
	Scaling	2	.409	.205	3.989	.112	.666
	FS	1	5.189	5.189	93.682	.050*	.879
	CS	1	5.978	5.978	116.500	.000**	.967
	MDT × CS	1	.386	.386	7.524	.050*	.653
	MDT × Scaling × CS	2	.546	.273	5.318	.050*	.727
Kitchenham	Method	2	28.695	14.348	94.024	.000**	.979
	MDT	1	.479	.479	7.549	.043*	.716
	Scaling	2	.244	.122	.799	.050*	.685
	FS	1	.006	.006	.038	.856	.009
	CS	1	.065	.065	.428	.549	.097
USPFT	Method	2	9.549	4.774	238.819	.000**	.992
	MDT	1	.870	.870	43.538	.003**	.916
	Scaling	2	1.908	.954	47.726	.002**	.960
	FS	1	2.050	2.050	102.564	.001**	.962
	CS	1	.145	.145	7.270	.050*	.645

FS × CS	1	1.476	1.476	73.829	.001**	.949
Scaling × FS	2	2.979	1.489	74.496	.001**	.974
Method × FS × CS	2	.545	.272	13.623	.016*	.872
Scaling × FS × CS	2	.340	.170	8.501	.036*	.810
Method × MDT × CS	2	.711	.356	17.789	.010**	.899
Method × MDT × FS	2	.477	.239	11.937	.021*	.857
Method × Scaling × FS	4	3.164	.791	39.567	.002**	.975
Method × Scaling × FS × CS	2	.587	.294	14.682	.014*	.880

**Significant at the 0.05 level (2-tailed)*

***Highly significant at the 0.01 level (2-tailed)*

4.6 Overall performance

Table 13 summarizes the testing results across of all experiments. The data shows that CBR and CART generally achieve the best overall performance in terms of average accuracy, followed by ANN, which could be better in terms of PRED(0.25) or MdBRE. All the methods seem very close to each other under the error metric PRED(0.25) and MdBRE in most cases.

Table 13

Overall averages and standard deviations of error values

Dataset		ISBSG		Desharnais		Kitchenham		USPFT	
Testing Error Metrics	ML Methods	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.
MBRE	CBR	3.582	4.358	14.1	8.78	2.51	0.340	180	109
	ANN	4.964	4.842	51.1	18.4	12.5	8.99	341	177
	CART	3.159	2.938	25.8	17.6	2.61	0.373	150	50.8
PRED	CBR	0.482	0.0637	0.471	0.0484	0.230	0.0515	0.611	0.0911
	ANN	0.554	0.0566	0.510	0.0397	0.202	0.0423	0.388	0.104
	CART	0.508	0.0417	0.483	0.0399	0.226	0.0206	0.465	0.128
MdBRE	CBR	0.271	0.0474	0.274	0.0423	0.792	0.196	0.0676	0.118
	ANN	0.223	0.0324	0.251	0.0292	0.930	0.265	0.480	0.216
	CART	0.245	0.0340	0.266	0.0215	0.789	0.0583	0.305	0.0735

To further analyze the error metrics in testing phase, we draw out the boxplots of MBRE, PRED(0.25), and MdBRE on three datasets in Fig. 2, Fig. 3, Fig. 4 and Fig. 5. The boxplots confirm the findings from Table 13. CBR and CART appear to obtain either lowest medians or shortest inter-quartiles range among all methods in the plots of MBRE, PRED and MdBRE. In general, ANN has the largest inter-quartiles range, especially happens when MBRE is the testing error metric, and what is more, it has long ranges. The performance of CBR is slightly influenced by outliers, so the mean results in Table 13 cannot generally outperform CART. Although boxplots are useful graphical tool for visually comparing predictions, they cannot statistically confirm

whether one technique is significant better than another. Therefore, we perform the significant tests on the error values.

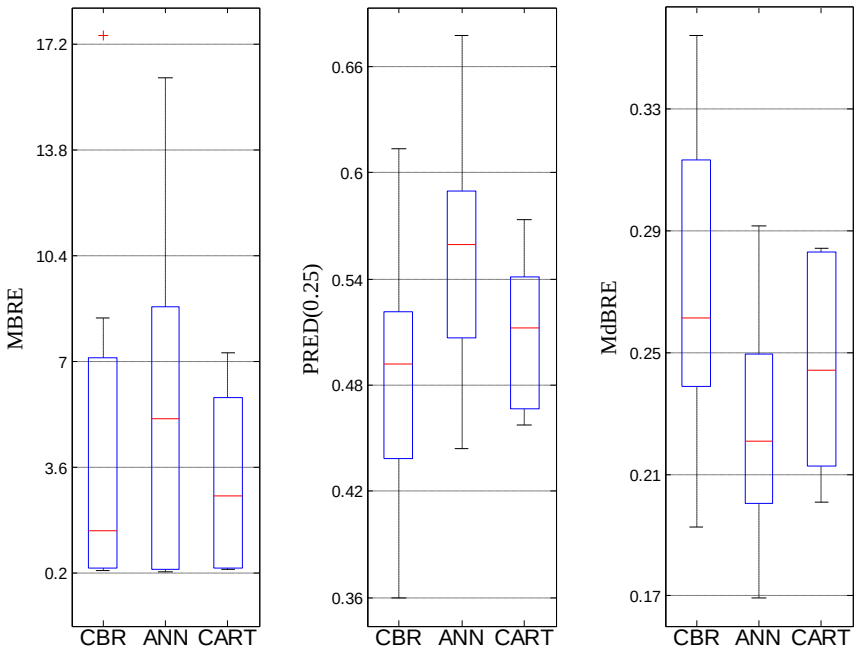


Fig. 2 Boxplots of error metric values of ISBSG dataset

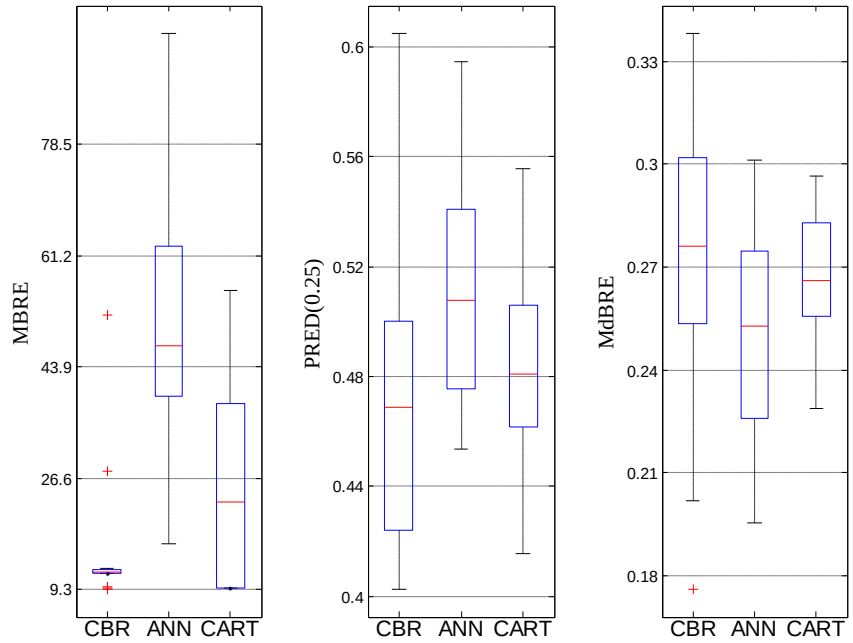


Fig. 3 Boxplots of error metric values of Desharnais dataset

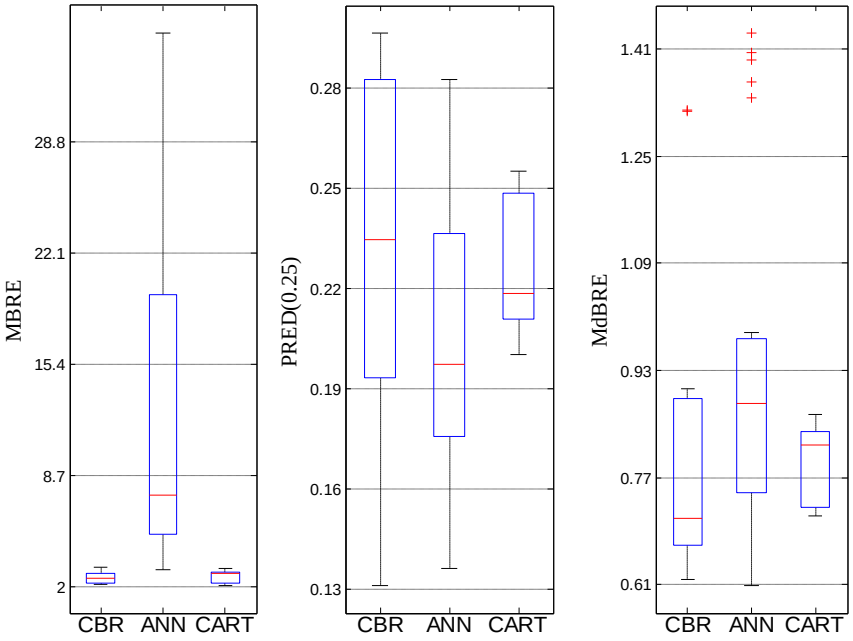


Fig. 4 Boxplots of error metric values of Kitchenham dataset

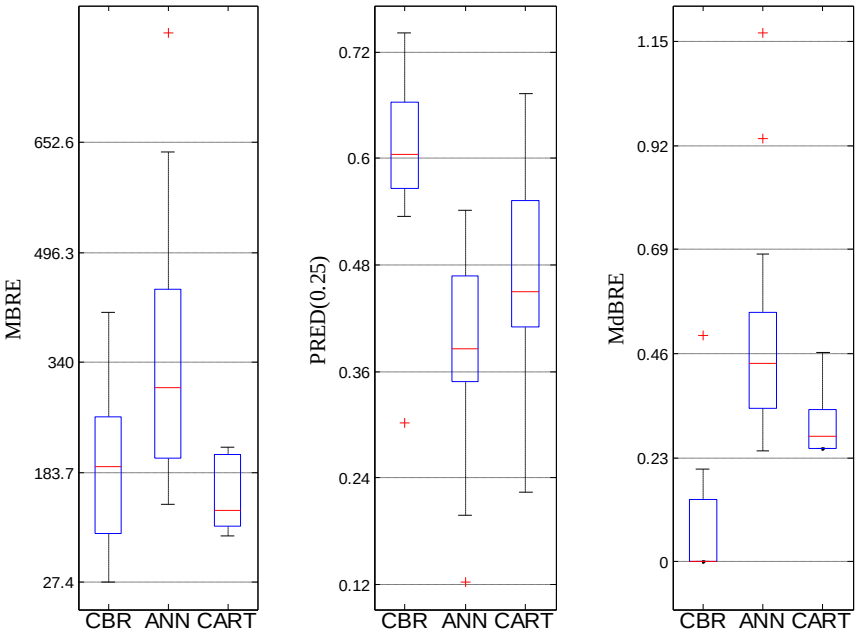


Fig. 5 Boxplots of error metric values of USPFT dataset

Because the boxplots show that the results distributions are highly skewed and do not satisfy the requirements of the classical *t*-test, the assumption free test: Wilcoxon

sign-rank test (significance level $\alpha = 0.05$ and $\alpha = 0.01$ with two tails) is chosen to test the method pairs. The *p*-values of the Wilcoxon tests are presented in Table 14. The entry in the upper triangle range contains the *p*-value of the test between methods located in its corresponding row and column. From the results of three testing error metrics in Fig. 2 and Fig. 4, we can tell that CBR is generally better than ANN, but not always better than CART. The only obvious exception happens in the plots of PRED and MdBRE of Desharnais and ISBSG dataset as shown in Fig. 2 & 3. ANN appears to perform better than the other two ML methods in terms of both PRED and MdBRE. Observing from Table 13, CBR and ANN are obviously significantly different in terms of all testing error metrics. For Desharnais dataset, CBR and CART are significantly different in terms of MBRE; for ISBSG and USPFT datasets, they are only different in terms of PRED and MdBRE; and for Kitchenham dataset, they are very the same. These findings imply that CBR and CART could be the best choice among the three ML methods on all datasets. CBR obviously outperforms CART on Desharnais and USPFT datasets. On ISBSG and Kitchenham datasets, the performance of CART and CBR are not significantly different.

Table 14

Results of Wilcoxon sign-rank tests

Dataset		ISBSG		Desharnais		Kitchenham		USPFT	
Testing Error Metrics	ML Methods	ANN	CART	ANN	CART	ANN	CART	ANN	CART
MBRE	CBR	.457	.819	.000**	.049*	.000**	.209	.000**	.049*
	ANN	-	.010**	-	.000**	-	.000**	-	.000**
PRED	CBR	.000**	.003**	.014*	.413	.134	.697	.000**	.002**
	ANN	-	.000**	-	.050*	-	.015*	-	.043*
MdBRE	CBR	.000**	.000**	.050*	.317	.110	.607	.000**	.000**
	ANN	-	.003**	-	.050*	-	.046*	-	.002**

**Highly significant at the 0.01 level (2-tailed)*

***Significant at the 0.05 level (2-tailed)*

One drawback of the above analysis is that the testing error metrics used are all based on BRE. Because BRE is relative error metric [94], we also consider Absolute Residuals (AR) as the alternative testing error metric for further analysis. We apply two-tailed Wilcoxon signed-rank test (α=0.05) to investigate the significance of

methods under BRE and AR across three random data splits on 24 generated datasets for preprocessing (totally 24 × 3 = 72). Table 15 presents the summary of the testing results. The symbols '=', '<', and '>' represent the relationships of 'equal', 'smaller than', and 'larger than' between the ML methods located in column and in row. The table entry denotes how many significance tests report the relationship represented by the symbol. The entries in the upper triangle range represent the results from significant tests on AR values whilst the entries in the lower triangle range are for the tests on BRE values. For instance, the entry with value '16' under column 'CBR' and the sub-column '>' means that there are totally 16 out 72 tests reported that ANN is significantly larger than CBR in terms of BRE in ISBSG dataset.

Table 15

Summary of Wilcoxon sign-rank tests for AR and BRE of each experiment

ML	CBR			ANN			CART			CBR			ANN			CART		
Methods	=	<	>	=	<	>	=	<	>	=	<	>	=	<	>	=	<	>
Dataset	ISBSG									Kitchenham								
CBR	-	-	-	53	19	0	68	3	1	-	-	-	60	12	0	66	5	1
ANN	56	0	16	-	-	-	59	0	13	60	0	12	-	-	-	55	3	14
CART	66	2	4	57	15	0	-	-	-	68	1	3	58	14	0	-	-	-
	Desharnais									USPFT								
CBR	-	-	-	67	5	0	63	9	0	-	-	-	51	21	0	59	13	0
ANN	68	0	4	-	-	-	67	0	5	34	5	33	-	-	-	62	1	9
CART	61	0	11	61	11	0	-	-	-	48	3	21	56	11	5	-	-	-

It is seen that both CBR and CART stand a fair chance to outperform ANN on ISBSG dataset. For instance, 16/72 significance tests report that ANN's BRE values are larger than those of CBR and 15/72 tests confirm that CART's BRE values are smaller than those of ANN. For AR, 19/72 significance tests confirm that CBR's AR values are smaller than those of ANN and 3/72 tests confirm that CBR's AR are smaller than those of CART. Note that the result of CART method in Table 13 is influenced by the outliers. On Desharnais dataset, the three methods are generally equal. In specific, CBR is slightly better than ANN or CART on both datasets. On Kitchenham dataset, the results are similar to that on ISBSG dataset. On USPFT dataset, CBR obviously outperforms ANN and CART. For instance, 21/72 tests confirm that CART's BRE values are larger than those of CBR. In summary, CBR appears to be a better alternative compared with ANN and CART in terms of all kinds

combinations of DP methods used in this study.

At the last, we analyze the performance of the combinations of DP techniques. We rank the DP combination according to the value of testing error metrics. From the experiment design, there are 24 processed datasets. Therefore, we have 24 sets of results for MBRE, PRED and MdBRE. We only keep the best 33% of the ranked results, *i.e.* 8 minimal MBREs, 8 maximal PREDs and 8 minimal MdBREs, and their corresponding DP methods. As expected, we did not find an overall dominant DP combination for all the four datasets and three ML methods. However, we do obtain results that could provide meaningful suggestion for other research works. The results are organized in Table 16.

Table 16

Summary of recommended DP combinations under different datasets and ML methods

Dataset	ML Methods	CBR					ANN					CART				
		Error value in order	DP combination				Error value in order	DP combination				Error value in order	DP combination			
			MDT	*Sca	FS	CS		MDT	*Sca	FS	CS		MDT	*Sca	FS	CS
ISBSG	MBRE	.2808	**1**	**3**	**2**	**2**	.2500	**1**	**3**	**2**	**1**	.3128	**1**	**1**	**2**	**2**
		.2986	**1**	**3**	**2**	**1**	.2517	**1**	**3**	**2**	**2**	.3128	**1**	**2**	**2**	**2**
		.3214	**1**	**1**	**2**	**1**	.2548	**1**	**1**	**2**	**1**	.3128	**1**	**3**	**2**	**2**
		.3214	**1**	**2**	**2**	**1**	.2843	**1**	**2**	**2**	**2**	.3506	**1**	**1**	**2**	**1**
		.3425	**1**	**1**	**2**	**2**	.3050	1	1	2	2	.3506	**1**	**2**	**2**	**1**
		.3425	**1**	**2**	**2**	**2**	.3145	1	2	2	1	.3506	**1**	**3**	**2**	**1**
		.3563	1	1	1	1	.3491	1	2	1	2	.3590	1	1	1	1
		.3613	1	2	1	1	.3516	1	1	1	2	.3590	1	2	1	1
	PRED	.6133	**1**	**3**	**2**	**2**	.6773	**1**	**3**	**2**	**2**	.5733	**1**	**1**	**2**	**2**
		.6000	**1**	**3**	**2**	**1**	.6333	**1**	**1**	**2**	**1**	.5733	**1**	**2**	**2**	**2**
		.5381	**2**	**3**	**1**	**1**	.6266	**1**	**3**	**2**	**1**	.5733	**1**	**3**	**2**	**2**
		.5333	1	1	2	1	.6147	2	1	1	1	.5466	**1**	**1**	**2**	**1**
		.5333	**1**	**2**	**2**	**1**	.6132	2	2	1	2	.5466	**1**	**2**	**2**	**1**
		.5224	2	1	1	1	.5920	**1**	**2**	**2**	**2**	.5466	**1**	**3**	**2**	**1**
		.5200	**1**	**1**	**2**	**2**	.5874	2	1	1	2	.5358	2	1	1	1
		.5200	**1**	**2**	**2**	**2**	.5710	2	3	1	2	.5358	2	2	1	1
	MdBRE	.1925	**1**	**3**	**2**	**2**	.1693	**1**	**3**	**2**	**2**	.2007	**1**	**1**	**2**	**2**
		.1992	**1**	**3**	**2**	**1**	.1784	**1**	**3**	**2**	**1**	.2007	**1**	**2**	**2**	**2**
		.2256	**1**	**1**	**2**	**1**	.1794	**1**	**1**	**2**	**1**	.2007	**1**	**3**	**2**	**2**
		.2256	**1**	**2**	**2**	**1**	.1902	2	2	1	2	.2065	**1**	**1**	**2**	**1**
		.2285	2	3	1	1	.1906	2	1	1	1	.2065	**1**	**2**	**2**	**1**
		.2380	2	1	1	1	.1995	**1**	**2**	**2**	**2**	.2065	**1**	**3**	**2**	**1**
		.2399	**1**	**1**	**2**	**2**	.2008	2	1	1	2	.2188	2	1	1	2
		.2399	**1**	**2**	**2**	**2**	.2032	2	3	2	1	.2188	2	2	1	2
Desharnais	MBRE	9.380	**2**	**3**	**1**	**2**	16.36	**2**	**3**	**2**	**2**	9.528	**2**	**1**	**1**	**2**
		9.590	1	1	2	2	27.07	**1**	**1**	**2**	**2**	9.528	**2**	**2**	**1**	**2**
		9.660	1	2	1	2	28.25	**1**	**1**	**2**	**1**	9.528	**2**	**3**	**1**	**2**
		9.668	1	1	1	2	34.32	**2**	**1**	**2**	**2**	9.539	2	1	2	2
		11.83	2	3	2	2	38.47	2	1	1	1	9.539	2	2	2	2
		11.85	**2**	**2**	**1**	**2**	38.84	1	1	1	1	9.539	2	3	2	2
		11.85	2	2	2	1	39.92	1	2	2	2	9.647	1	1	2	2
		11.86	2	1	1	1	42.90	**2**	**2**	**1**	**2**	9.647	1	2	2	2
	PRED	.6049	2	3	1	1	.5948	**2**	**3**	**2**	**2**	.5555	2	1	2	1
		.5308	**2**	**2**	**1**	**2**	.5822	**2**	**1**	**2**	**2**	.5555	2	2	2	1
		.5194	1	3	2	1	.5528	2	1	2	1	.5555	2	3	2	1
		.5185	**2**	**3**	**1**	**2**	.5472	1	2	2	1	.5185	2	1	1	1
		.5064	1	1	2	1	.5451	**1**	**1**	**2**	**1**	.5185	2	2	1	1
		.5064	1	3	2	2	.5433	1	2	2	2	.4935	**2**	**3**	**1**	**2**
		.4938	2	2	2	2	.5387	**2**	**2**	**1**	**2**	.4935	**2**	**2**	**1**	**2**
		.4935	1	1	1	1	.5376	**1**	**1**	**2**	**2**	.4935	**2**	**1**	**1**	**2**
	MdBRE	.1760	2	3	1	1	.1953	**2**	**1**	**2**	**2**	.2287	2	1	2	1
		.2018	1	3	2	2	.2039	**2**	**3**	**2**	**2**	.2287	2	2	2	1

		.2131	**2**	**2**	**1**	**2**	.2196	2	1	2	1	.2287	2	3	2	1
		.2438	1	3	2	1	.2227	**1**	**1**	**2**	**1**	.2475	2	1	1	1
		.2484	1	1	2	1	.2252	1	2	2	2	.2475	2	2	1	1
		.2484	**2**	**3**	**1**	**2**	.2256	1	2	2	1	.2635	**2**	**3**	**1**	**2**
		.2583	1	1	1	1	.2262	**1**	**1**	**2**	**2**	.2635	**2**	**2**	**1**	**2**
		.2583	1	3	1	1	.2296	**2**	**2**	**1**	**2**	.2635	**2**	**1**	**1**	**2**
	MBRE	2.118	1	3	2	2	3.040	1	1	1	1	2.060	**2**	**1**	**1**	**1**
		2.156	**1**	**1**	**2**	**1**	3.317	**2**	**3**	**2**	**2**	2.060	**2**	**2**	**1**	**1**
		2.156	1	2	2	1	3.480	**1**	**2**	**2**	**2**	2.060	**2**	**3**	**1**	**1**
		2.163	**2**	**3**	**1**	**1**	4.624	2	2	1	1	2.129	**2**	**1**	**2**	**1**
		2.165	**2**	**3**	**2**	**1**	4.722	1	2	1	2	2.129	**2**	**2**	**2**	**1**
		2.201	**1**	**3**	**1**	**1**	5.084	**1**	**1**	**2**	**1**	2.129	**2**	**3**	**2**	**1**
		2.203	**1**	**3**	**2**	**1**	5.274	1	2	2	1	2.316	1	1	1	1
		2.237	**1**	**1**	**2**	**2**	5.499	1	2	1	1	2.316	1	2	1	1
	PRED	.2962	**1**	**1**	**2**	**2**	.2824	2	3	1	2	.2551	**2**	**1**	**2**	**1**
		.2962	1	2	2	2	.2711	1	3	2	2	.2551	2	1	2	2
		.2896	**2**	**3**	**1**	**1**	.2619	**1**	**2**	**2**	**2**	.2551	**2**	**2**	**2**	**1**
Kitchenham		.2896	**2**	**3**	**2**	**1**	.2533	2	1	2	1	.2551	**2**	**3**	**2**	**1**
		.2888	**1**	**3**	**1**	**1**	.2470	**2**	**3**	**2**	**2**	.2551	2	3	2	2
		.2888	**1**	**3**	**2**	**1**	.2407	**1**	**1**	**2**	**1**	.2482	**2**	**1**	**1**	**1**
		.2758	2	3	1	2	.2318	2	1	2	2	.2482	**2**	**2**	**1**	**1**
		.2592	**1**	**1**	**2**	**1**	.2313	1	1	1	1	.2482	**2**	**3**	**1**	**1**
	MdBRE	.6170	2	3	1	2	.6088	2	1	2	1	.7130	1	1	2	2
		.6175	**2**	**3**	**1**	**1**	.6201	**1**	**2**	**2**	**2**	.7130	1	3	2	2
		.6253	**1**	**3**	**1**	**1**	.6406	**2**	**3**	**2**	**2**	.7220	2	1	2	2
		.6253	**1**	**3**	**2**	**1**	.6560	**1**	**1**	**2**	**1**	.7220	2	3	2	2
		.6293	**2**	**3**	**2**	**1**	.7019	2	1	2	2	.7246	**2**	**1**	**2**	**1**
		.6682	1	3	2	2	.7342	2	3	1	2	.7246	**2**	**2**	**2**	**1**
		.6690	**1**	**1**	**2**	**1**	.7598	1	3	2	2	.7246	**2**	**3**	**2**	**1**
		.6690	**1**	**1**	**2**	**2**	.7608	1	2	2	1	.7400	**2**	**1**	**1**	**1**
	MBRE	27.44	**1**	**1**	**2**	**1**	138.1	2	3	2	1	93.47	**2**	**1**	**1**	**1**
		27.45	**1**	**1**	**2**	**2**	149.3	**1**	**1**	**2**	**2**	93.47	**2**	**2**	**1**	**1**
		27.47	**1**	**3**	**2**	**1**	169.2	2	1	1	1	93.47	**2**	**3**	**1**	**1**
		34.80	2	1	2	2	177.0	1	3	2	1	103.6	**1**	**1**	**1**	**2**
		52.47	2	1	2	1	177.7	1	2	1	1	103.6	**1**	**2**	**1**	**2**
		95.83	2	3	2	2	186.3	2	3	2	2	103.6	**1**	**3**	**1**	**2**
		95.87	2	3	2	1	221.8	1	2	2	2	110.3	1	1	2	2
		97.96	**1**	**3**	**2**	**2**	228.6	**1**	**1**	**2**	**1**	110.3	1	2	2	2
	PRED	.7413	**1**	**3**	**2**	**1**	.5410	**1**	**1**	**2**	**1**	.6724	**1**	**1**	**1**	**2**
		.7241	**1**	**1**	**2**	**1**	.5375	1	3	1	1	.6724	**1**	**2**	**1**	**2**
		.7241	**1**	**3**	**2**	**2**	.5162	1	3	1	2	.6724	**1**	**3**	**1**	**2**
USPFT		.7068	**1**	**1**	**2**	**2**	.5006	**1**	**1**	**2**	**2**	.5789	**2**	**1**	**1**	**1**
		.7068	1	2	2	1	.4844	2	1	2	2	.5789	**2**	**2**	**1**	**1**
		.6724	1	2	2	2	.4837	1	1	1	1	.5789	**2**	**3**	**1**	**1**
		.6551	1	1	1	1	.4521	2	1	2	1	.5263	2	1	1	2
		.6447	2	1	1	2	.4362	1	2	1	2	.5263	2	2	1	2
	MdBRE	0	1	1	1	2	.2447	1	3	1	1	.2495	**1**	**1**	**1**	**2**
		0	**1**	**1**	**2**	**1**	.2467	**1**	**1**	**2**	**1**	.2495	**1**	**2**	**1**	**2**
		0	**1**	**1**	**2**	**2**	.2536	1	3	1	2	.2495	**1**	**3**	**1**	**2**
		0	1	2	1	2	.2681	**1**	**1**	**2**	**2**	.2498	**2**	**1**	**1**	**1**
		0	1	2	2	1	.3023	2	1	2	2	.2498	2	1	1	2
		0	1	2	2	2	.3081	1	1	1	1	.2498	**2**	**2**	**1**	**1**
		0	**1**	**3**	**2**	**2**	.3704	2	1	2	1	.2498	2	2	1	2
		0	**1**	**3**	**2**	**1**	.3838	1	2	1	1	.2498	**2**	**3**	**1**	**1**

MDT, 1 – Listwise deletion; 2 – Mean imputation
**Sca: Scaling, 1 – [0, 1], 2 – [-1, 1], 3 – Null*
FS, 1 – FSS, 2 – Null;
CS, 1 – BSS, 2 – Null.

All the repeated DP combinations inside a particular data under a specific ML method are marked in bold. For example, in dataset of ISBSG, under CBR method, the DP combination of 1-3-2-2 (Listwise deletion, Null scaling, without FSS for FS or BSS for CS) is marked in bold since it exists in MBRE, PRED and MdBRE simultaneously.

For specific ML method, we can see that 1-1-2-1 is a stable choice for ANN since it shows up in bold for all the three datasets used in this study. For CBR, the

performance of DP combinations for datasets varies. Only 1-1-2-1, 1-3-2-1 and 1-1-2-2 have stable performances on ISBSG, Kitchenham and USPFT datasets. As for CART, the results further identifies that it is totally immune to [0, 1] scaling or [-1, 1] scaling. Whatever the scaling scheme is, once the rest DP combination is the same, the prediction results of CART are the same as well. The performance of DP combinations in CART varies a lot, only 2-X-1-1 (X is denoted as [0, 1] or [-1, 1] or Null) performs relatively stable for both USPFT and Kitchenham datasets and the corresponding estimation accuracy is promising. We could only suggest using the DP combination of 1-X-2-2 for ISBSG dataset and 2-X-1-2 for Desharnais dataset when SCE is based on CART algorithm.

Further observation from Table 16 provides more useful knowledge of implementation of DP combinations. The DP combination should be carefully used according to the characteristics of data and the estimation method as well. All the recommendations in terms of using data preprocessing for corresponding datasets or ML-based methods in the context of SCE are listed in Table 17. Notice that in Table 16, only the best DP combinations are selected from the rankings in Table 16. The criterion for selection is based on the ranking of the average value of MBRE, -PRED, and MdMER, shown in the 3rd column of Table 17. Note that the recommended DP combinations in Table 17 are denoted in the same way Table 16 presents. The ANOVA shows that certain DPs and DP combinations could be overall significantly effective to SCE. The result showing in Table 17 further shows which DP combinations are the most positively effective to different datasets and different ML methods. It gives us the answer to RQ5 in terms of DP techniques selection.

Table 17
Recommended DP Combinations

Dataset	ML method	(MBRE-PRED+MdBRE)/3	Recommended DP combinations
ISBSG	CBR	-0.0467	1-3-2-2
	ANN	-0.0854	1-3-2-2
	CART	-0.0199	1-X*-2-2
Desharnais	CBR	3.04	2-3-1-2
	ANN	5.32	2-3-2-2
	CART	3.10	2-X*-1-2

Kitchenham	CBR	0.830	2-3-1-1
	ANN	1.24	2-3-2-2
	CART	0.851	2-X*-1-1
USPFT	CBR	8.91	1-1-2-1
	ANN	49.7	1-1-2-2
	CART	31.0	2-X*-1-1

X - [0, 1] or [-1, 1] or Null (scaling has no impact on CART)*

The results in Table 16 and Table 17 show that for Desharnais, Kitchenham and USPFT datasets, all the best DP combinations include MI. For the dataset with high percentage of missing values, e.g. ISBSG dataset, MI may not provide accurate estimation of the missing data. In Desharnais, Kitchenham and USPFT datasets, the issue of missing values is not as critical as that in ISBSG data. MI may be a good choice for ANN based SCE in these three datasets.

Further observation from Table 17 shows that most of the best DP combinations contain Null scaling scheme, except on USPFT data, the scaling scheme of [0, 1] is preferred. Observing that the most SCE studies adopt [0, 1] scheme intuitively, we recommend researchers to consider Null scaling alternatively to see if results are improved. The scaling scheme of [-1, 1] is less suggested since it does not appear in any recommended DP combinations.

Although applying FS for ANN could be overall beneficial on Desharnais data as Table 10 presents, the result in Table 17 shows that all the best DP combinations for ANN method in four datasets do not include FSS for FS. Similar situation happens when applying CS for ANN. Weigh repeatedly, we suggest not conducting FS or CS for ANN based SCE.

CS should be very carefully regarded when using it into a DP combination. Observing from Table 16, all the well-performed DP combinations for ISBSG, USPFT and Kitchenham data under CART contains BSS for CS. But if we take a look at the results in Table 11, the prediction accuracy of CART based estimation in Kitchenham data is in average reduced by CS. From the results, we found that certain DP combinations with BSS for CS could seriously lower the accuracy of CART based SCE; therefore, the mean testing error in Table 11 becomes more unsatisfied. The

similar situation applies to ISBSG data as well. The result in Table 17 shows that most of the best DP combinations in four datasets do not include BSS for CS. In conclusion, CS might only be useful for better prediction results of CBR or CART on specific datasets, and it should be conducted more carefully.

5. Threats to Validity

The threats to validity could be distributed into four groups in this study, including conclusion, internal, construct, and external validity. The conclusion validity is related to the ability to draw significant correct conclusions; in regard to which, we carefully applied the statistical tests, showing statistical significance for the obtained results. Moreover, we used medium-sized datasets to mitigate the threats related to the number of observations composing the datasets.

As this study focuses on the structured investigations on DP techniques, we discuss the internal validity in terms of threats to our experiments designs. There are several aspects: firstly, different numbers of pre-defined parameters are considered for each ML method in this study. For example, CBR has three parameters while CART has only one parameter. We observed that CBR is generally not significantly different from CART and both of them are more accurate than ANN method. However CBR's accuracy may be due to the extra parameters of CBR. Therefore, the sensitivity analysis of parameters is of interest to identify which parameter is important and how does the parameter affect the prediction performances. In addition, more sophisticated parameter tuning schemes can be explored because ML methods performances heavily depend on these parameters and the cross-validation plus grid search scheme is time-consuming to select parameters with large ranges. Moreover, this study chooses MBRE to guide parameter optimization of ML methods. However, MBRE is only one type of balanced relative error metric, other types of balanced error metrics such as *Mean Square Error* (MSE) could be considered as the error metric for parameter selection.

The construct validity refers to the agreement between a theoretical concept and a specific measurement. It has to establish correct operational measures for the concepts being studied. As to the assessment of different combinations of DP techniques, we made use of three performance measures and four public datasets in our study. All the performance measures are used with statistical test as suggested by Kitchenham et al. [98]. The datasets in our study have been previously used in many other empirical studies carried out to evaluate effort estimation.

The external validity represents the possibility of generalizing the findings of this study. The threats to external validity are as follows. Firstly, the limited datasets cause some difficulties to generalize our conclusions. The ISBSG dataset is very large and regarded as a benchmarking dataset in SCE literature, but it is multi-organizational in nature and with heterogeneous properties. There are no commonly accepted data refinement criteria to the knowledge of the authors. Nevertheless, we refer to works of [37, 66, 85] and refine the ISBSG data. In real world estimation procedures, many project managers prefer to use the data from one company to ensure data consistency. Therefore, our results based on the four datasets might be difficult to generalize to those estimations obtained from single and homogeneous company data. Recently, some research works [99, 100] have been conducted to investigate to what extent a cross-company cost model can be successfully employed to estimate single company projects. Nevertheless, the experiment design proposed in our study can be easily extended to other single company datasets.

In addition, missing data is a common situation in software engineering datasets. Many studies [29, 31, 39, 74] have proposed different imputation techniques to recover missing data by estimating replacement values. But our study is mainly focusing on the LD and MI. More advanced methods of imputation could be adopted to deal with missing data. Moreover, limited DP techniques and ML methods are explored in our study. There might be some difficulties to extend our conclusions to other ML methods to select appropriate DP scheme. Furthermore, in the experiment

the 3-fold cross-validation scheme, though it is the most widely used one in the literature, might not produce sufficiently stable estimation of the prediction accuracy. More advanced scheme as 10-fold cross-validation or even leave-one-out cross-validation could be considered. The side effect of these schemes is mainly the high computational expense.

6. Conclusion and Future Works

Data preprocessing is a fundamental stage of ML method and has large impact on the accuracy of ML methods. However, there is still lack of systematic study in the SCE context for DP techniques despite of their importance. In this study, a structured literature survey of DP techniques is first conducted. Subsequently, a systematic empirical study is conducted to analyze the effectiveness of the four DP techniques, *i.e.* MDT, Scaling, FS and CS. In addition, the interactions between the preprocessing techniques and ML methods' predictive accuracies are also studied. ANOVA test is conducted to quantify the significance of each preprocessing technique and the interactions between them and ML methods. In the end, the recommendation of using DP combination for different datasets and different ML methods is summarized (See Table 17). For example, the best DP combination for Kitchenham data in the context of CBR based SCE is MI, Null scaling, FSS for FS, and BSS for CS. Seven findings are noteworthy:

1. The effectiveness of an ML method used for SCE (e.g. CBR, ANN and CART) can be significantly altered by single DP steps and their combinations (especially, the MDT, scaling, FS), which further extends the findings found by Keung et al. [35].

2. The performance of DP methods is generally dependent on the characteristics of data and the ML methods.

3. The percentage of missingness inside data is an important matter when we consider using MDT for ML-based SCE. Compared with MI, LD is preferred for datasets with a large number of missing values in terms of improving prediction accuracy and robustness. For datasets with a small number of missing values, ANN is less sensitive to MI and CART could benefit from MI.

4. No need to use [0, 1] or [-1, 1] scaling for CART based SCE. The scaling scheme of Null scaling could be better than [0, 1] under certain circumstance. Scaling appears to be less significant to large and highly-skewed dataset.

5. FS is more appropriate for CBR based SCE rather than for ANN or CART. For ANN and CART, the impact of FS is dependent on the characteristics of data, such as the number of features. CS could be effective, but it is not recommended using CS for ANN.

6. ANN performs worse than other two ML methods in our study (e.g. CBR and CART), similar to the results obtained by Keung et al. [35]. CBR and CART could be better methods for SCE according to our experiments. CART is negatively impacted by FS and more sensitive to CS. An appropriate strategy of DP is necessary for CART.

7. The selection of DP combination is dependent on datasets and ML methods. We suggest the SCE research works that utilize any ML method consider fine-tuning the DP techniques so that the performance of ML method can be improved. This finding conveys a meaningful message to the majority of the completed or ongoing SCE research works where the DP techniques have been or are being used without sufficient attentions.

As for the future works, 1) more extensive experiments on various datasets will be considered to generalize the findings of this study; 2) more advanced ML methods can be involved in the experiment, for example the more complicated tree model of CART, such as multiple additive regression trees (MART) and multivariate adaptive

regression splines (MARS); 3) use unbiased novel performance measure, such as standardized accuracy [101], for comparison of DPs.

References

[1] J. Wen, S. Li, Z. Lin, Y. Hu, C. Huang, Systematic literature review of machine learning based software development effort estimation models, Inf Softw Technol, 54 (2012) 41-59.

[2] R.T. Hughes, Expert judgement as an estimating method, Inf Softw Technol, 38 (1996) 67-75.

[3] M. Jørgensen, A review of studies on expert estimation of software development effort, J Syst Softw, 70 (2004) 37-60.

[4] B.W. Boehm, Software engineering economics, IEEE Trans Softw Eng, SE-10 (1984) 4-21

[5] L.H. Putnam, W. Myers, Measures for excellence: reliable software on time, within budget, Prentice Hall Professional Technical Reference, 1991.

[6] A. Bou, D. Ho, L. Fernando, Towards an early software estimation using log-linear regression and a multilayer perceptron model, J Syst Softw, 86 (2013) 144-160.

[7] K. Srinivasan, D. Fisher, Machine learning approaches to estimating software development effort, IEEE Trans Softw Eng, 21 (1995) 126-137.

[8] A. Heiat, Comparison of artificial neural network and regression models for estimating software development effort, Inf Softw Technol, 44 (2002) 911-922.

[9] M. Shepperd, C. Schofield, Estimating software project effort using analogies, IEEE Trans Softw Eng, 23 (1997) 736-743.

[10] C. Mair, G. Kadoda, M. Lefley, K. Phalp, C. Schofield, M. Shepperd, S. Webster, Investigation of machine learning based prediction systems, J Syst Softw, 53 (2000) 23-29.

[11] I.F. de Barcelos Tronto, J.D.S. da Silva, N. Sant'Anna, An investigation of artificial neural networks based prediction systems in software project management, J Syst Softw, 81 (2008) 356-367.

[12] G.R. Finnie, G.E. Wittig, J.M. Desharnais, A comparison of software effort estimation techniques: Using function points with neural networks, case-based reasoning and regression models, J Syst Softw, 39 (1997) 281-289.

[13] Y.F. Li, M. Xie, T.N. Goh, A study of mutual information based feature selection for case based reasoning in software cost estimation, Expert Syst Appl, 36 (2009) 5921-5931.

[14] E. Kocaguneli, S. Member, T. Menzies, Exploiting the essential assumptions of analogy-based effort estimation, IEEE Trans Softw Eng, 38 (2012) 425-439.

[15] M. Azzeh, D. Neagu, P.I. Cowling, Analogy-based software effort estimation using fuzzy numbers, J Syst Softw, 84 (2011) 270-284.

[16] E. Kocaguneli, T. Menzies, J.W. Keung, Kernel methods for software effort estimation: Effects of different kernel functions and bandwidths on estimation accuracy, Empirical Softw Eng, 18 (2013) 1-24.

[17] N.H. Chiu, S.J. Huang, The adjusted analogy-based software effort estimation based on similarity distances, J Syst Softw, 80 (2007) 628-640.

[18] S.J. Huang, N.H. Chiu, Optimization of analogy weights by genetic algorithm for software effort estimation, Inf Softw Technol, 48 (2006) 1034-1045.

[19] J. Li, G. Ruhe, Analysis of attribute weighting heuristics for analogy-based software effort estimation method AQUA+, Empirical Softw Eng, 13 (2008) 63-96.
[20] A.L.I. Oliveira, P.L. Braga, R.M.F. Lima, M.L. Cornélio, GA-based method for feature selection and parameters optimization for machine learning regression applied to software effort estimation, Inf Softw Technol, 52 (2010) 1155-1166.
[21] S. Zhang, C. Zhang, Q. Yang, Data preparation for data mining, Appl Artif Intell, 17 (2003) 375-381.
[22] B. Twala, M. Cartwright, Ensemble missing data techniques for software effort prediction, Intell Data Analysis, 14 (2010) 299-331.
[23] L.L. Minku, X. Yao, A principled evaluation of ensembles of learning machines for software effort estimation, in: the 7th International Conference on Predictive Models in Software Engineering (PROMISE'11), Banff, Canada, 2011, pp. 1-10.
[24] M. Azzeh, Software effort estimation based on optimized model tree, in: the 7th International Conference on Predictive Models in Software Engineering (PROMISE'11), Banff, Canada, 2011.
[25] M. Shepperd, G. Kadoda, Comparing software prediction techniques using simulation, IEEE Trans Softw Eng, 27 (2001) 1014-1022.
[26] J.W. Keung, B.A. Kitchenham, D.R. Jeffery, Analogy-X: Providing statistical inference to analogy-based software cost estimation, IEEE Trans Softw Eng, 34 (2008) 471-484.
[27] K. Vinay Kumar, V. Ravi, M. Carr, N. Raj Kiran, Software development cost estimation using wavelet neural networks, J Syst Softw, 81 (2008) 1853-1867.
[28] Y.F. Li, M. Xie, T.N. Goh, A study of project selection and feature weighting for analogy based software cost estimation, J Syst Softw, 82 (2009) 241-252.
[29] K. Strike, K.E. Emam, N. Madhavji, Software cost estimation with incomplete data, IEEE Trans Softw Eng, 27 (2001) 890-908.
[30] J. Van Hulse, T.M. Khoshgoftaar, A comprehensive empirical evaluation of missing value imputation in noisy software measurement data, J Syst Softw, 81 (2008) 691-708.
[31] I. Myrtveit, E. Stensrud, U.H. Olsson, Analyzing data sets with missing data: An empirical evaluation of imputation methods and likelihood-based methods, IEEE Trans Softw Eng, 27 (2001) 999-1013.
[32] P. Sentas, L. Angelis, Categorical missing data imputation for software cost estimation by multinomial logistic regression, J Syst Softw, 79 (2006) 404-414.
[33] Y.-S. Seo, D.-H. Bae, On the value of outlier elimination on software effort estimation research, Empirical Softw Eng, 18 (2013) 659-698.
[34] M. Tsunoda, T. Kakimoto, A. Monden, K.-i. Matsumoto, An empirical evaluation of outlier deletion methods for analogy-based cost estimation, in: the 7th International Conference on Predictive Models in Software Engineering (PROMISE '11), Banff, Canada, 2011, pp. 1-10.
[35] J. Keung, E. Kocaguneli, T. Menzies, Finding conclusion stability for selecting the best effort predictor in software effort estimation, Automated Software Engineering, 20 (2013) 543-567.
[36] M. Azzeh, D. Neagu, P.I. Cowling, Fuzzy grey relational analysis for software effort estimation, Empirical Softw Eng, 15 (2009) 60-90.
[37] N. Mittas, L. Angelis, LSEbA: least squares regression and estimation by analogy in a semi-parametric model for software cost estimation, Empirical Softw Eng, 15 (2010) 523-555.
[38] J. Li, G. Ruhe, A. Al-Emran, M.M. Richter, A flexible method for software effort estimation by analogy, Empirical Softw Eng, 12 (2007) 65-106.

[39] Q. Song, M. Shepperd, A new imputation method for small software project data sets, J Syst Softw, 80 (2007) 51-62.
[40] L. Angelis, I. Stamelos, A simulation tool for efficient analogy based cost estimation, Empirical Softw Eng, 5 (2000) 35–68.
[41] E. Kocaguneli, S. Member, T. Menzies, On the value of ensemble effort estimation, IEEE Trans Softw Eng, 38 (2012) 1403-1416.
[42] A. Corazza, S.D. Martino, F. Ferrucci, C. Gravino, F. Sarro, E. Mendes, How effective is Tabu search to configure support vector regression for effort estimation?, in: the 6th International Conference on Predictive Models in Software Engineering (PROMISE'10), Timisoara, Romania, 2010, pp. 1-10.
[43] A. Corazza, S. Di Martino, F. Ferrucci, C. Gravino, E. Mendes, Applying support vector regression for Web effort estimation using a cross-company dataset, in: the 3rd International Symposium on Empirical Software Engineering and Measurement (ESEM'09), Lake Buena Vista, Florida, USA, 2009, pp. 191-202.
[44] N. Mittas, M. Athanasiades, L. Angelis, Improving analogy-based software cost estimation by a resampling method, Inf Softw Technol, 50 (2008) 221-230.
[45] J. Li, A. Al-Emran, G. Ruhe, Impact analysis of missing values on the prediction accuracy of analogy-based software effort estimation method AQUA, in: the 1st International Symposium on Empirical Software Engineering and Measurement (ESEM'07), Madrid, Spain, 2007, pp. 126-135.
[46] E. Mendes, A comparison of techniques for web effort estimation, in: the 1st International Symposium on Empirical Software Engineering and Measurement (ESEM'07), Madrid, Spain, 2007, pp. 334 - 343.
[47] J. Keung, Empirical evaluation of analogy-X for software cost estimation, in: the 2nd ACM-IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM'08), Kaiserslautern, Germany, 2008, pp. 294-296.
[48] F. Ferrucci, E. Mendes, F. Sarro, Web effort estimation: the value of cross-company data set compared to single-company data set, in: the 8th International Conference on Predictive Models in Software Engineering (PROMISE'12), Lund, Sweden, 2012, pp. 29-38.
[49] P.C. Pendharkar, G.H. Subramanian, J.A. Rodger, A probabilistic model for predicting software development effort, IEEE Trans Softw Eng, 31 (2005) 615-624.
[50] M. Auer, A. Trendowicz, B. Graser, E. Haunschmid, S. Biffl, Optimal project feature weights in analogy-based cost estimation: Improvement and limitations, IEEE Trans Softw Eng, 32 (2006) 83-92.
[51] T. Menzies, A. Butcher, D. Cok, A. Marcus, L. Layman, F. Shull, B. Turhan, T. Zimmermann, Local versus global lessons for defect prediction and effort estimation, IEEE Trans Softw Eng, 39 (2013) 822-834.
[52] A. Brady, T. Menzies, Case-based reasoning vs Parametric models for software quality optimization, in: the 6th International Conference on Predictive Models in Software Engineering (PROMISE'10), Timisoara, Romania, 2010, pp. 1-10.
[53] E. Kocaguneli, T. Menzies, How to Find Relevant Data for Effort Estimation?, in: the 5th International Symposium on Empirical Software Engineering and Measurement (ESEM'11), Banff, Canada, 2011, pp. 255-264.
[54] Y.F. Li, M. Xie, T.N. Goh, A study of the non-linear adjustment for analogy based software cost estimation, Empirical Softw Eng, 14 (2009) 603-643.
[55] R. Borges, T. Menzies, Learning to change projects, in: the 8th International Conference on

Predictive Models in Software Engineering (PROMISE'12), Lund, Sweden, 2012, pp. 11-18.
[56] E. Kocaguneli, T. Menzies, J. Hihn, B.H. Kang, Size doesn't matter? On the value of software size features for effort estimation, in: the 8th International Conference on Predictive Models in Software Engineering (PROMISE'12), Lund, Sweden, 2012, pp. 89-98.
[57] Q. Liu, W.Z. Qin, R. Mintram, M. Ross, Evaluation of preliminary data analysis framework in software cost estimation based on ISBSG R9 Data, Softw Qual J, 16 (2008) 411-458.
[58] V. Khatibi Bardsiri, D.N.A. Jawawi, S.Z.M. Hashim, E. Khatibi, A PSO-based model to increase the accuracy of software development effort estimation, Softw Qual J, 21 (2013) 501-526.
[59] A. Corazza, S. Martino, F. Ferrucci, C. Gravino, F. Sarro, E. Mendes, Using tabu search to configure support vector regression for effort estimation, Empirical Softw Eng, 18 (2013) 506-546.
[60] J. Li, G. Ruhe, Decision support analysis for software effort estimation by analogy, in: the 3rd International Conference on Predictor Models in Software Engineering (PROMISE'07), Minneapolis, MN, USA, 2007, pp. 6-16.
[61] L.L. Minku, X. Yao, Ensembles and locality: Insight on improving software effort estimation, Inf Softw Technol, 55 (2013) 1512-1528.
[62] M. Azzeh, A replicated assessment and comparison of adaptation techniques for analogy-based effort estimation, Empirical Softw Eng, 17 (2012) 90-127.
[63] C.-J. Hsu, C.-Y. Huang, Comparison of weighted grey relational analysis for software effort estimation, Softw Qual J, 19 (2010) 165-200.
[64] E. Kocaguneli, T. Menzies, J. Keung, D. Cok, R. Madachy, Active learning and effort estimation: Finding the essential content of software effort estimation data, IEEE Trans Softw Eng, 39 (2013) 1040-1053.
[65] N. Mittas, L. Angelis, Ranking and clustering software cost estimation models through a multiple comparisons algorithm, IEEE Trans Softw Eng, 39 (2013) 537-551.
[66] C. Lopez-Martin, C. Isaza, A. Chavoya, Software development effort prediction of industrial projects applying a general regression neural network, Empirical Softw Eng, 17 (2011) 738-756.
[67] A. Bakir, B. Turhan, A.B. Bener, A new perspective on data homogeneity in software cost estimation: A study in the embedded systems domain, Softw Qual J, 18 (2009) 57-80.
[68] A. Bakir, B. Turhan, A. Bener, A comparative study for estimating software development effort intervals, Softw Qual J, 19 (2011) 537-552.
[69] N. Ramasubbu, R.K. Balan, Overcoming the challenges in cost estimation for distributed software projects, in: the 34th International Conference on Software Engineering (ICSE'12), Zurich, Switzerland, 2012, pp. 91-101.
[70] M.V. Kosti, N. Mittas, L. Angelis, Alternative methods using similarities in software effort estimation, in: the 8th International Conference on Predictive Models in Software Engineering (PROMISE'12), Lund, Sweden, 2012, pp. 59-68.
[71] D. Rodríguez, M.A. Sicilia, E. García, R. Harrison, Empirical findings on team size and productivity in software development, J Syst Softw, 85 (2012) 562-570.
[72] Q. Song, M. Shepperd, M. Cartwright, A short note on safest default missingness mechanism assumptions, Empirical Softw Eng, 10 (2005) 235-243.
[73] J. Moses, M. Farrow, Assessing variation in development effort consistency using a data source with missing data, Softw Qual J, 13 (2005) 71-89.
[74] J. Van Hulse, T.M. Khoshgoftaar, Incomplete-case nearest neighbor imputation in software measurement data, Inform Sciences, 259 (2014) 596-610.

[75] D. Michie, D.J. Spiegelhalter, C.C. Taylor, Machine learning, neural and statistical classification, (1994).
[76] S.-J. Huang, N.-H. Chiu, Optimization of analogy weights by genetic algorithm for software effort estimation, Inf Softw Technol, 48 (2006) 1034-1045.
[77] Z. Chen, T. Menzies, D. Port, B. Boehm, Feature subset selection can improve software cost estimation accuracy, in: the 2005 International Conference on Predictor Models in Software Engineering (PROMISE'05), St. Louis, MO, USA, 2005, pp. 1-6.
[78] S. Das, Filters, wrappers and a boosting-based hybrid for feature selection, in: the 18th International Conference on Machine Learning (ICML'01), Williamstown, MA, USA, 2001, pp. 74-81.
[79] E. Mendes, I. Watson, C. Triggs, N. Mosley, S. Counsell, A comparative study of cost estimation models for web hypermedia applications, Empirical Softw Eng, 8 (2003) 163-196.
[80] ISBSG, The ISBSG Development & Enhancement project data in: ISBSG (Ed.), http://www.isbsg.org, 2013.
[81] I. Guyon, A. Elisseeff, An introduction to variable and feature selection, J Mach Learn Res, 3 (2003) 1157-1182.
[82] H. Peng, F. Long, C. Ding, Feature selection based on mutual information: Criteria of max-dependency, max-relevance, and min-redundancy, IEEE Trans Pattern Anal Mach Intell, 27 (2005) 1226-1238.
[83] P.L. Braga, A.L.I. Oliveira, S.R.L. Meira, A GA-based feature selection and parameters optimization for support vector regression applied to software effort estimation, in: the 23rd Annual ACM Symposium on Applied Computing (SAC'08), Fortaleza, Ceara, Brazil, 2008, pp. 1788-1792.
[84] C. Kirsopp, M. Shepperd, Case and feature subset selection in case-based software project effort prediction, in: M. Bramer, A. Preece, F. Coenen (Eds.) Research and Development in Intelligent Systems XIX, Springer London, 2003, pp. 61-74.
[85] M. Fernández-Diego, F. González-Ladrón-de-Guevara, Potential and limitations of the ISBSG dataset in enhancing software engineering research: A mapping review, Inf Softw Technol, XXX.
[86] N. Mittas, L. Angelis, Visual comparison of software cost estimation models by regression error characteristic analysis, J Syst Softw, 83 (2010) 621-637.
[87] N. Mittas, L. Angelis, A permutation test based on regression error characteristic curves for software cost estimation models, Empirical Softw Eng, 17 (2011) 34-61.
[88] Y.-S. Seo, D.-H. Bae, R. Jeffery, AREION: Software effort estimation based on multiple regressions with adaptive recursive data partitioning, Inf Softw Technol, 55 (2013) 1710-1725.
[89] J.M. Desharnais, Analyse statistique de la productivitie des projets informatique a partie de la technique des point des foncti\ on, in, University of Montreal, 1989.
[90] M. Jørgensen, M. Shepperd, A systematic review of software development cost estimation studies, IEEE Trans Softw Eng, 33 (2007) 33-53.
[91] F. Walkerden, R. Jeffery, Empirical study of analogy-based software effort estimation, Empirical Softw Eng, 4 (1999) 135-158.
[92] A.R. Gray, S.G. MacDonell, A comparison of techniques for developing predictive models of software metrics, Inf Softw Technol, 39 (1997) 425-437.
[93] Y.F. Li, M. Xie, T.N. Goh, Adaptive ridge regression system for software cost estimating on multi-collinear datasets, J Syst Softw, 83 (2010) 2332-2343.
[94] T. Foss, E. Stensrud, B. Kitchenham, I. Myrtveit, A simulation study of the model evaluation criterion MMRE, IEEE Trans Softw Eng, 29 (2003) 985-995.

[95] I. Myrtveit, E. Stensrud, M. Shepperd, Reliability and validity in comparative studies software prediction models, IEEE Trans Softw Eng, 31 (2005) 380-391.
[96] M. Ochodek, J. Nawrocki, K. Kwarciak, Simplifying effort estimation based on Use Case Points, Inf Softw Technol, 53 (2011) 200-213.
[97] M.a. Ahmed, I. Ahmad, J.S. AlGhamdi, Probabilistic size proxy for software effort prediction: A framework, Inf Softw Technol, 55 (2013) 241-251.
[98] B. Kitchenham, E. Mendes, Why comparative effort prediction studies may be invalid, in: the 5th International Conference on Predictor Models in Software Engineering (PROMISE'09), ACM, Vancouver, Canada, 2009, pp. 4.
[99] E. Mendes, C. Lokan, Replicating studies on cross- vs single-company effort models using the ISBSG Database, Empirical Softw Eng, 13 (2008) 3-37.
[100] E. Mendes, S. Di Martino, F. Ferrucci, C. Gravino, Cross-company vs. single-company web effort models using the Tukutuku database: An extended study, J Syst Softw, 81 (2008) 673-690.
[101] M. Shepperd, S. MacDonell, Evaluating prediction systems in software project estimation, Inf Softw Technol, 54 (2012) 820-827.