Data Mining Techniques for Software Effort Estimation: A Comparative Study

Karel Dejaeger, Wouter Verbeke, David Martens, and Bart Baesens

Abstract—A predictive model is required to be accurate and comprehensible in order to inspire confidence in a business setting. Both aspects have been assessed in a software effort estimation setting by previous studies. However, no univocal conclusion as to which technique is the most suited has been reached. This study addresses this issue by reporting on the results of a large scale benchmarking study. Different types of techniques are under consideration, including techniques inducing tree/rule-based models like M5 and CART, linear models such as various types of linear regression, nonlinear models (MARS, multilayered perceptron neural networks, radial basis function networks, and least squares support vector machines), and estimation techniques that do not explicitly induce a model (e.g., a case-based reasoning approach). Furthermore, the aspect of feature subset selection by using a generic backward input selection wrapper is investigated. The results are subjected to rigorous statistical testing and indicate that ordinary least squares regression in combination with a logarithmic transformation performs best. Another key finding is that by selecting a subset of highly predictive attributes such as project size, development, and environment related attributes, typically a significant increase in estimation accuracy can be obtained.

Index Terms—Data mining, software effort estimation, regression.

1 Introduction

Resource planning is considered a key issue in a production environment. In the context of a software developing company, the different resources are, among others, computing power and personnel. In recent years, computing power has become a subordinate resource for software developing companies as it doubles approximately every 18 months, thereby costing only a fraction compared to the late 1960s. Personnel costs are, however, still an important expense in the budget of software developing companies. In light of this observation, proper planning of personnel effort is a key aspect for these companies. Due to the intangible nature of the product "software," software developing companies are often faced with problems estimating the effort needed to complete a software project [1]. There has been strong academic interest in this topic, assisting the software developing companies in tackling the difficulties experienced to estimate software development effort [2]. In this field of research, the required effort to develop a new project is estimated based on historical data from previous projects. This information can be used by management to improve the planning of personnel, to make more accurate tendering bids, and to evaluate risk factors [3]. Recently, a number of studies evaluating different techniques have been published. The results of these studies are not univocal and are often highly technique and data set dependent. In this paper, an overview of the existing literature is presented. Furthermore, 13 techniques, representing different kinds of models, are investigated. This selection includes tree/rule-based models (M5 and CART), linear models (ordinary least squares regression with and without various transformations, ridge regression (RiR), and robust regression (RoR)), nonlinear models (MARS, least squares support vector machines, multilayered perceptron neural networks (NN), radial basis function (RBF) networks), and a lazy learning-based approach which does not explicitly construct a prediction model, but instead tries to find the most similar past project. Each technique is applied to nine data sets within the domain of software effort estimation. From a comprehensibility point of view, a more concise model (i.e., a model with less inputs) is preferred. Therefore, the impact of a generic backward input selection approach is assessed.

The remainder of this paper is structured as follows: First, Section 2 presents an overview of the literature concerning software effort estimation. Then, in Section 3 the different techniques employed in the study are discussed. Section 4 reflects upon the data sets, evaluation criteria, and the statistical validation of the study. In Section 5, we apply the techniques as well as the generic backward input selection schema and discuss the results. The paper is concluded by Section 6 providing general conclusions and topics for future research.

- *K. Dejaeger and W. Verbeke are with the Department of Decision Sciences and Information Management, Katholieke Universiteit Leuven, Naamsestraat 69, Leuven B-3000, Belgium. E-mail: {Karel.Dejaeger, Wouter.Verbeke}@econ.kuleuven.be.*
- *D. Martens is with the Faculty of Applied Economics, University of Antwerp, Prinsstraat 13, Antwerp B-2000, Belgium. E-mail: David.Martens@ua.ac.be.*
- *B. Baesens is with the Department of Decision Sciences and Information Management, Katholieke Universiteit Leuven, Naamsestraat 69, Leuven B-3000, Belgium and with the School of Management, University of Southampton, Highfield Southampton, SO17 1BJ, United Kingdom. E-mail: Bart.Baesens@econ.kuleuven.be.*

Manuscript received 4 June 2010; revised 29 Oct. 2010; accepted 5 Jan. 2011; published online 15 June 2011.
Recommended for acceptance by A.A. Porter.
For information on obtaining reprints of this article, please send e-mail to: tse@computer.org, and reference IEEECS Log Number TSE-2010-06-0170.
Digital Object Identifier no. 10.1109/TSE.2011.55.

2 Related Research

In the field of software effort estimation, the effort required to develop a new software project is estimated by taking the

TABLE 1
Overview of the Cocomo I Multipliers

EM_i	Description	Impact
acap	Analysts capability	Positive impact: Increase of this factor results in a decreased effort
pcap	Programmers capability	
aexp	Application experience	
modp	Modern programming practices	
tool	Use of software tools	
vexp	Virtual machine experience	
lexp	Language experience	
sced	Schedule constraint	Convex relation
stor	Main memory constraint	Negative impact: Increase of this factor results in an increased effort
data	Data base size	
time	Time constraint for cpu	
turn	Turnaround time	
virt	Machine volatility	
cplx	Process complexity	
rely	Required software reliability	

details of the new project into account. The specific project is then compared to a historical data set (i.e., a set of past projects) containing measurements of relevant metrics (e.g., size, language used, and experience of development team) and the associated development effort.

The first approaches to estimate software development effort were introduced in the late 1960s [4], and relied on expert judgment. In these cases, a domain expert applies his or her prior experience to come up with an estimation of the needed effort. A number of different variations exist, e.g., *Delphi expert estimation*, in which several experienced developers formulate an independent estimate and the median of these estimates is used as the final effort estimation [5]. While still widely used in companies, an expert-driven approach has the disadvantage of lacking an objective underpinning. Furthermore, the domain expert is key to the estimation process, inducing additional risks. In [6], an overview of the literature concerning expert estimation was presented, concluding that "There are situations where expert estimates are more likely to be more accurate... Similarly, there are situations where the use of models may reduce large situational or human biases."

During the last 30 years, a number of formal models for software effort estimation have been proposed such as Cocomo [7], Cocomo II [8], SLIM [9], and Function Points Analysis [10]. These models have some advantages, providing a formulaic underpinning of software effort estimation [11]. Hence, these models allow for a number of analyses to be performed upon the obtained results [7, chapter 3]. Companies applying formal models during the estimation process often opt for a Constructive Cost Model (Cocomo model). The Cocomo I model takes the following form:

$$\text{Effort} = a \times \text{Size}^b \prod_{i=1}^{15} \text{EM}_i,$$

where a and b are two factors that can be set depending on the details of the developing company and EM_i is a set of effort multipliers, see Table 1. As data sets typically do not contain sufficient projects to calibrate all parameters, only a and b are adapted to reflect the development environment using a local calibration approach. Data for this model are collected making use of specific questionnaires which are filled in by the project manager. This data collection approach require a considerable effort from the business. Also, it should be noted that the Cocomo I model is already somewhat outdated as, e.g., new software development trends such as outsourcing and multiplatform development are not taken into account by the model. A newer version of the Cocomo model exists [8], but the data on which this model was built is not publicly available.

More recently, formal models are being superseded by a number of data intensive techniques originating from the data mining literature [2]. These include various regression techniques which result in a linear model [12], [13], nonlinear approaches like neural networks [12], tree/rule-based models such as CART [14], [15], and lazy learning strategies (also referred to as case-based reasoning (CBR)) [16]. Data mining techniques typically result in objective and analyzable formulas which are not limited to a specific set of attributes, as is the case with formal models such as Cocomo I. Due to these strong elements, data mining approaches are also adopted on a regular basis in numerous other research and business applications such as credit scoring [17] and customer churn prediction [18]. A number of studies have assessed the applicability of data mining techniques to software effort estimation. However, most of these studies evaluate only a limited number of modeling techniques on a particular, sometimes proprietary, data set which naturally constrains the generalizability of the observed results. Some of these studies also lack a proper statistical testing of the obtained results or evaluate models on the same data as used to build the models [19].

A nonexhaustive overview of the literature concerning the use of various machine learning approaches for software effort estimation is presented in Table 2. This table summarizes the applied modeling techniques, the data sets that are used, and the empirical setup for a number of studies. As can be seen from Table 2, a large number of modeling techniques have been applied in search for the most suitable technique for software effort estimation, both in terms of accuracy and comprehensibility.

For example, Finnie et al. [12] compared Artificial Neural Networks (ANN) and Case-Based Reasoning to Ordinary Least Squares regression (OLS regression). It was found that both artificial intelligence models (ANN and CBR) outperformed OLS regression and thus can be adequately used for software effort estimation. However, these results were not statistically tested.

Briand et al. [14], while performing a comparison between OLS regression, stepwise ANOVA, CART, and CBR, found that case-based learning achieved the worst results, while CART performed best; however, the difference was not found to be statistically significant. In a follow-up study using the same techniques on a different data set, different results were obtained, i.e., stepwise ANOVA and OLS regression performed best [15].

Shepperd and Schofield [20] reported that CBR outperforms regression, yet in a study by Myrtveit and Stensrud [21], these results were not confirmed. However, Shepperd and Schofield used a different regression approach (without a log transformation) than the latter study and opted not to split the data set in a training and test set.

TABLE 2
Literature Overview of the Application of Data Mining Approaches for Software Effort Estimation

Authors	Title & *Journal*	Year	What?	Techniques	Data set & *# proj. - # attr.*	Metrics - Empirical setup
K. Srinivasan, D. Fisher	Machine Learning Approaches to Estimating Software Development Effort *IEEE Transactions on Software Engineering*	1995	Two alternative data mining techniques are compared to formal models	Artificial neural networks, CART	Kemerer *15 proj. - 6 attr.* Cocomo81 *63 proj. - 16 attr.*	MMRE, R^2 - Holdout
M. Shepperd, C. Schofield	Estimating Software Project Effort Using Analogies *IEEE Transactions on Software Engineering*	1997	Investigation of data mining techniques as an alternative to formal models	OLS regression, case based reasoning	DPS database *24 proj. - 5 attr.* Desharnais *77 proj. - 9 attr.* Finnish *38 proj. - 29 attr.* Kemerer *15 proj. - 2 attr.* Mermaid *28 proj. - 17 attr.* Telecom data sets *18-33 proj. - 1-13 attr.*	MMRE, $Pred_{25}$ - Cross validation
G. Wittig, G. Finnie	Estimating software development effort with connectionist models *Information and Software Technology*	1997	Analysis of backpropagation neural networks for software effort prediction	Artificial neural networks	Simulated data *1000 proj. - 3 attr.* Desharnais *81 proj. - 9 attr.*	MMRE, $Pred_{25}$ - Holdout
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 *The Journal of Systems and Software*	1997	Comparison of three different estimation techniques using function points as a size estimate	OLS regression, artificial neural networks, case based reasoning	ASMA data set *299 proj. - 1 attr.*	MMRE, $Pred_{25}$ - Holdout
L. Briand, K. El Emam, D. Surmann, I. Wieczorek, K. Maxwell	An Assessment and Comparison of Common Software Cost Estimation Modeling techniques *21st International Conference on Software Engineering (ICSE '99)*	1999	Investigation of the performance of various software effort estimation techniques both for company-specific and multi-organizational data	OLS regression, ANOVA, CART, case based reasoning	Experience *206 proj. - 19 attr.*	MMRE, MdMRE, $Pred_{25}$ - Cross validation and matched pairs Wilcoxon signed rank test
I. Myrtveit, E. Stensrud	A Controlled Experiment to Assess the Benefits of Estimation with Analogy and Regression Models *IEEE Transactions on Software Engineering*	1999	Experiment of software effort estimation by experts using case based reasoning and regression techniques as estimation tool	OLS regression, case based reasoning	COTS data set *48 proj. - 10 attr.*	MMRE, MdMRE - Paired t-test and Wilcoxon rank sum test
L. Briand, T. Langley, I. Wieczorek	A replicated assessment and Comparison of Common Software Cost Modeling Techniques *22nd International Conference on Software Engineering (ICSE '00)*	2000	In line with their previous study, a similar set of techniques are applied to a different data set	OLS regression, ANOVA, CART, case base reasoning	ESA *166 proj. - 10 attr.*	MdMRE, $Pred_{25}$- Cross validation and matched pairs Wilcoxon signed rank test
C. Burgess, M. Lefley	Can genetic programming improve software effort estimation? A comparative evaluation *Information and Software Technology*	2001	Evaluation of genetic programming algorithms for software effort prediction	Artificial neural networks, case based reasoning, genetic programming	Desharnais *81 proj. - 9 attr.*	MMRE, BMMRE, $Pred_{25}$, correlation, AMSE - Holdout
K. Strike, K. El Emam, N. Madhavji	Software Cost Estimation with Incomplete Data *IEEE Transactions on Software Engineering*	2001	Study on the effects of missing data in the field of software effort prediction	OLS regression	Experience *206 proj. - 16 attr.*	MdMRE, Pred, R^2_{adj} - Holdout
B. Kitchenham, S. L. Pfleeger, B. McColl, S. Eagan	An empirical study of maintenance and development accuracy *The Journal of Systems and Software*	2002	Comparison of regression techniques using a real-life data set from a single company	OLS regression, median regression	CSC data set *144 proj. - 10 attr.*	MMRE, Pred - Holdout and paired t-test
A. Heiat	Comparison of artificial neural network and regression models for estimating software development effort *Information and Software Technology*	2002	Comparison of the performance of neural networks versus regression approaches	OLS regression, artificial neural networks, RBF networks	DPS database *24 proj. - 5 attr.* Kemerer *15 proj. - 6 attr.* Hallmark *28 proj. - ? attr.*	MMRE, R^2 - Holdout and paired t-test
P. Sentas, L. Angelis, I. Stamelos, G. Bleris	Software productivity and effort prediction with ordinal regression *Information and Software Technology*	2005	Analysis of a novel regression technique for software effort prediction	OLS regression, ordinal regression	Cocomo81 *63 proj. - 22 attr.* Maxwell *62 proj. - 14 attr.* ISBSG R7 *52 proj. - 10 attr.*	MMRE, $Pred_{25}$ - Holdout
T. Menzies, Z. Chen, J. Hihn, K. Lum	Selecting Best Practices for Effort Estimation *IEEE Transactions on Software Engineering*	2006	Investigation of deviations exhibited by different techniques for software estimation	OLS regression	Cocomo81 *63 proj. - 16 attr.* NASA *93 proj. - 16 attr.* COCOMOII *161 proj. - 18 attr.*	MMRE, $Pred_{30}$, Correlation - Holdout
M. Auer, A. Trendowicz, B. Graser, E. Hounschmid, S. Biffl	Optimal Project Feature Weights in Analogy-Based Cost Estimation: Improvement and Limitations *IEEE Transactions on Software Engineering*	2006	Investigation on the impact of different weighting schemata for case based reasoning	Case based reasoning	ESA *44 proj. - 11 attr.* Experience *43 proj. - 18 attr.* Desharnais *77 proj. - 9 attr.* DPS database *24 proj. - 6 attr.* Kemerer *15 proj. - 3 attr.*	MMRE, $Pred_{25}$, Variance - Cross validation
N.-H. Chiu, S.-J. Huang	The adjusted analogy-based software effort estimation based on similarity distances *The Journal of Systems and Software*	2007	The application of a genetic algorithm to derive an estimation based on the retrieved cases	OLS regression, artificial neural networks, CART, case based reasoning	DPS database *23 proj. - 5 attr.* Abran *21 proj. - 6 attr.*	MMRE, MdMRE, $Pred_{25}$ - Cross validation
J. Li, G. Ruhe, A. Al-Emran, M. Richter	A flexible method for software effort estimation by analogy *Empirical Software Engineering*	2007	Introduction of AQUA, a case based reasoning algorithm and subsequent comparison with other techniques	Case based reasoning	USP05 *197 proj. - 15 attr.* ISBSG R4 *2024 proj. - 18 attr.* Kemerer *15 proj. - 6 attr.* Leung02 *24 proj. - 16 attr.* Mends03 *34 proj. - 8 attr.*	MMRE, $Pred_{25}$, Strength, Support - Cross validation
S.-J. Huang, N.-H. Chiu, L.-W. Chen	Integration of the grey relational analysis with genetic algorithm for software effort estimation *European Journal of Operational Research*	2008	Building software effort estimation models using grey relational analysis to counteract incomplete observations	Artificial neural networks, CART, case based reasoning, Grey relational analysis	Cocomo81 *63 proj. - 16 attr.* DPS database *23 proj. - 5 attr.*	MMRE, $Pred_{25}$ - Holdout
K. Kumar, V. Ravi, M. Carr, N. Kiran	Software development cost estimation using wavelet neural networks *The Journal of Systems and Software*	2008	Investigation of wavelet neural networks in the context of software effort estimation	Artificial neural networks, RBF networks, wavelet neural networks, support vector machines	DPS database *23 proj. - 4 attr.* Abran *21 proj. - 6 attr.*	MMRE, MdMRE, $Pred_{25}$ - Cross validation
H. Park, S. Baek	An empirical validation of a neural network model for software effort estimation *Expert Systems with Applications*	2008	The evaluation of a neural network model for software effort estimation for function point based data sets	OLS regression, artificial neural networks	Korean IT data set *148 proj. - 18 attr.*	MMRE - Holdout
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 Systems with Applications*	2009	Application of a mutual information based filter approach to select attributes for case based reasoning	Case based reasoning	Desharnais *77 proj. - 8 attr.* Maxwell *62 proj. - 25 attr.*	MMRE, MdMRE, $Pred_{25}$ - Cross validation
S. Koch, J. Mitlöhner	Software project effort estimation with voting rules *Decision Support Systems*	2009	Application of social choice models to the domain of software effort estimation using a genetic algorithm to determine weights	Social choice models	DPS database *24 proj. - 5 attr.* ERP data set *31 proj. - 7 attr.* Cocomo81 *63 proj. - 15 attr.*	MMRE, MdMRE, $Pred_{25}$ - Cross validation

It should be noted that the results of different studies are often difficult to compare due to different empirical setup and data preprocessing, possibly leading to contradictory results. Hence, the issue of which modeling technique to use for software effort estimation remains an open research question. Another issue in software engineering is the fact that estimation techniques are typically applied to small data sets and/or data sets which are not publicly available, making studies not reproducible [22]. Additionally, studies often make comparisons based on only one or two data sets. Kitchenham and Mendes [19] note that "One of the main problems with evaluating techniques using one or two data sets is that no one can be sure that the specific data sets were not selected because they are the ones that favor the new technique."

Although a large variety of techniques are available, expert-driven estimation methods are still frequently applied in a business setting. Evidence from other domains suggests that both data mining and formal models could provide more accurate estimates than expert driven estimation methods. Often cited strong points of an analytical approach are consistency (provided with the same input, a model will always reach the same conclusion) and the fact that such models posses the ability to correctly assess the impact of different inputs [23]. This conjecture was, however, not confirmed by studies in the domain of software effort prediction [6]. Jørgensen stated that "The use of models, either alone or in combination with expert judgment may be particularly useful when 1) there are situational biases that are believed to lead to a strong bias toward overoptimism; 2) the amount of contextual information possessed by the experts is low; and 3) the models are calibrated to the organization using them." [24]. Other research confirmed that whether expert-driven methods perform significantly better or worse than an analytical oriented approach remains a point of debate [25], [26], [27].

3 Techniques

As mentioned in Section 2, a large number of different techniques have been applied to the field of software effort estimation. As the aim of the study is to assess which data mining techniques perform best to estimate software effort, the following techniques are considered[1]:

- Ordinary least squares regression.
- OLS regression with log transformation.
- OLS regression with Box Cox (BC) transformation.
- Robust regression.
- Ridge regression.
- Least median squares regression.
- MARS.
- CART.
- Model tree.
- Multilayered perceptron neural network.
- Radial basis function networks.
- Case-based reasoning.
- Least squares support vector machines.

These techniques were selected as their use has previously been illustrated in the domain of software effort prediction and/or promising results were obtained in other regression contexts. Due to the scale of the benchmarking experiment, computational cost was also taken into consideration in selecting the techniques, eliminating techniques characterized by high-computational loads.

The following notation is used throughout the paper. A scalar $x \in \mathbb{R}$ is denoted in normal script while a vector $\mathbf{x} \in \mathbb{R}^n$ is in boldface script. A vector is always a column vector unless indicated otherwise. A row vector is indicated as the transposed of the associated column vector, $\mathbf{x}'$. A matrix $\mathbf{X} \in \mathbb{R}^{N \times n}$ is in bold capital notation. $x_i(j)$ is an element of matrix $\mathbf{X}$ representing the value of the jth variable on the ith observation. N is used as the number of observations in a data set, while n represents the number of variables.

In the data sets, the target variable is effort in man-months or an equivalent measure. The actual effort of the ith software project is indicated as e_i, while the predicted effort is indicated as $\hat{e}_i$. In line with this notation, the task of estimating a continuous target can be defined as follows: Let $\mathrm{S} = \{(\mathbf{x}_i, e_i)\}_{i=1}^{N}$ be a training data set containing N observations, where $\mathbf{x}_i \in \mathbb{R}^n$ represents the characteristics of a software project and $e_i \in \mathbb{R}$ is the continuous target variable. An estimation model provides a mapping from the instances $\mathbf{x}_i$ to the continuous target, e_i: $f(\mathbf{x}_i) : \mathbb{R}^n \mapsto \hat{e}_i$.

In the following paragraphs, a short discussion of the included techniques is presented. A selection of regression techniques and data mining techniques (MARS, CART, and MLP; cfr infra) is applied to the Desharnais data set and illustrated in Figs. 1a, 1b, 1c, 1d, 1e, and 1f. A two dimensional representation that plots project size (Points-NonAdjust) against effort in man hours is used to allow for easier representation.

3.1 OLS

Arguably one of the oldest and most widely applied techniques for software effort estimation is Ordinary Least Squares regression. This well-documented technique fits a linear regression function to a data set containing a dependent, e_i, and multiple independent variables, $x_i(1)$ to $x_i(n)$; this type of regression is also commonly referred to as multiple regression. OLS regression assumes the following linear model of the data:

$$e_i = \mathbf{x}_i'\beta + b_0 + \epsilon_i,$$

where $\mathbf{x}_i'$ represents the row vector containing the values of the ith observation, $x_i(1)$ to $x_i(n)$. β is the column vector containing the slope parameters that are estimated by the regression, and b_0 is the intercept scalar. This intercept can also be included in the β vector by introducing an extra variable with a value of one for each observation. ϵ_i is the error associated with each observation. This error term is used to estimate the regression parameters, β, by minimizing the following objective function:

$$min \sum_{i=1}^{N} \epsilon_i^2,$$

thus obtaining the following estimate for β:

$$\hat{\beta} = (\mathbf{X}'\mathbf{X})^{-1}(\mathbf{X}'\mathbf{e}),$$

with $\mathbf{e}$ representing the column vector containing the effort and $\mathbf{X}$ an $N \times n$ matrix with the associated explanatory variables.

3.2 Log + OLS

Typically, both dependent and independent attributes in the field of software effort prediction can be heavily skewed, e.g., $\mathrm{skewness}(e)_{Desharnais} = 1.97$ and $\mathrm{skewness}(e)_{ESA} = 4.89$. Skewness, γ_s, is defined as

$$\gamma_s = \frac{\mu_3}{\sigma^3},$$

1. The techniques are implemented in Matlab, www.mathworks.com, and Weka, http://www.cs.waikato.ac.nz/ml/weka. Additionally, open source toolboxes were used in case of least squares support vector machines (LS-SVMlab, http://www.esat.kuleuven.be/sista/lssvmlab) and MARS (ARESLab, http://www.cs.rtu.lv/jekabsons/regression.html).

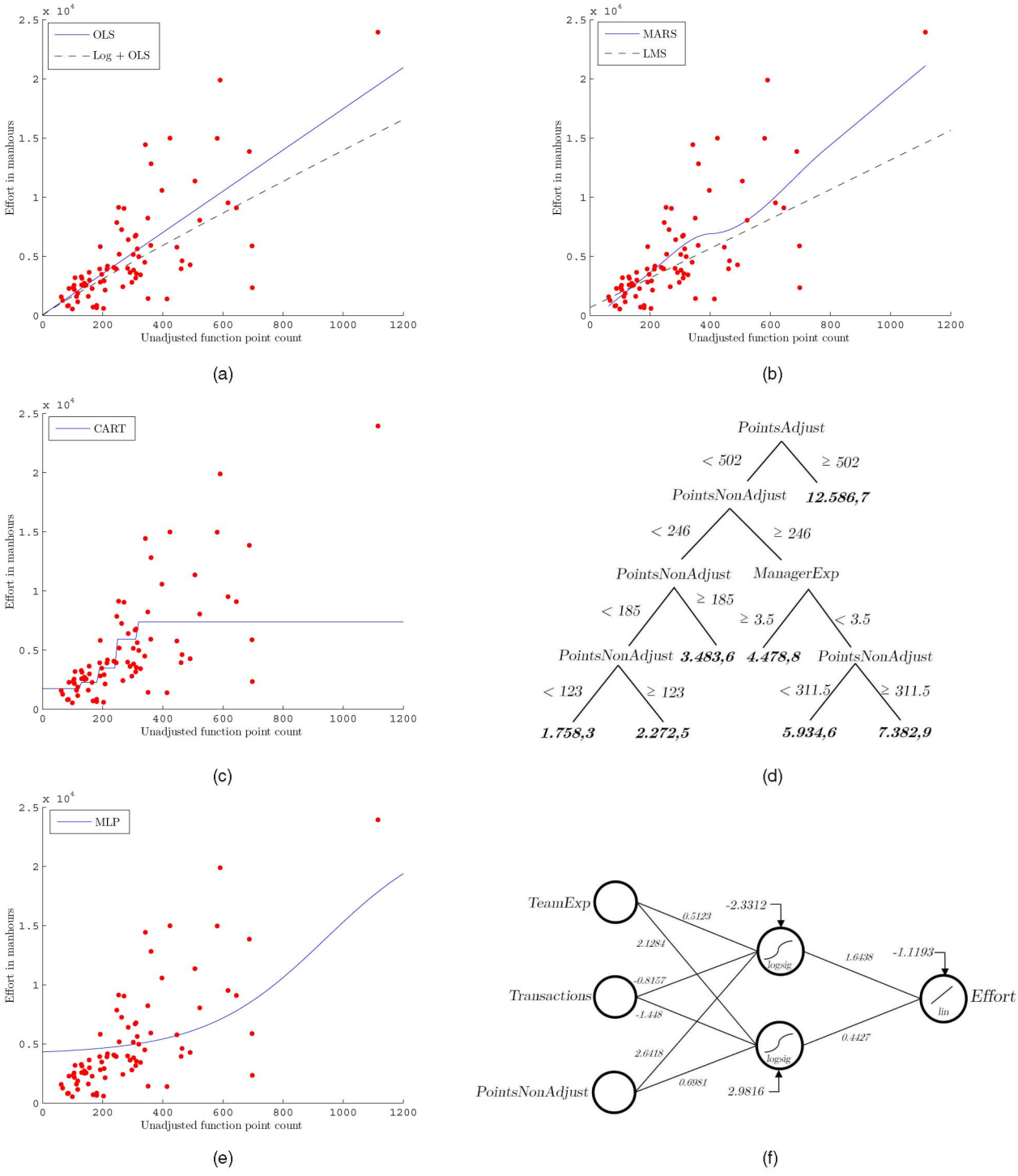


Fig. 1. Comparison of machine learning techniques on the Desharnais data set. (a) OLS regression with and without log transformation, (b) MARS and LMS. (c) and (e) The CART and MLP regression line, respectively. (d) and (f) The accompanying CART tree and neural network.

where μ_3 is the third moment of the mean and σ the standard deviation. A normal distribution has a skewness of zero while a positive (negative) skewness indicates a larger number of smaller (bigger) projects. By applying a log transformation to the data, the residuals of the regression model become more homoscedastic, and follow more closely a normal distribution [28]. This transformation is also used in previous studies [14], [15]. Both OLS and Log + OLS are illustrated in Fig. 1a.

3.3 BC + OLS

The Box Cox transformation is a power transformation which corrects for discontinuities, e.g., when the transformation parameter, γ, is zero [29]. The BC transformation is defined as

$$e_i^* = \begin{cases} \dfrac{(e_i^\gamma - 1)}{\gamma}, & \gamma \neq 0, \\ \log e_i, & \gamma = 0. \end{cases}$$

The transformation parameter is obtained by a maximum likelihood optimization. The BC transformation is an alternative to the log transformation serving similar goals. A BC transformation will also resolve problems related to nonnormality and heterogeneity of the error terms [30].

To the best of our knowledge, the BC transformation has not been previously used in the context of software effort estimation.

3.4 Robust Regression

RoR is an alternative to OLS regression with the advantage of being less vulnerable to the existence of outliers in the data [31]. RoR is an application of Iteratively Reweighted Least Squares (IRLS) regression in which the weights $\omega_i^{(t)}$ are iteratively set by taking the error terms of the previous iteration, $\epsilon_i^{(t-1)}$, into account.

In each iteration, RoR will minimize the following objective function:

$$min \sum_{i=1}^{N} \omega_i^2 \epsilon_i^2.$$

From this equation, it can be easily seen that OLS regression can be in fact considered as a special case of RoR [32].

Multiple possible weighting functions exist, the most commonly applied being Huber's weighting function and Tukey's bisquare weighting function. In this study, the Tukey's bisquare weighting function is used:

$$\omega_{bisquare} = \begin{cases} (1-\zeta^2)^2, & \text{if } |\zeta| < 1, \\ 0, & \text{otherwise.} \end{cases}$$

ζ is the normalized error term and is computed as a function of $\hat{\sigma}$, the estimated standard deviation of the error terms, and a tuning constant, τ, which penalizes for the distance to the regression function.

The first iteration consists of an OLS regression since the weights depend on the previous iteration.

RoR is a technique that has been previously applied in the field of software effort estimation [33].

3.5 Ridge Regression

RiR is an alternative regression technique that tries to address a potential problem with OLS in case of highly correlated attributes. OLS regression is known to be Best Linear Unbiased Estimator (BLUE) if a number of conditions are fulfilled, e.g., the fact that $\mathbf{X}'\mathbf{X}$ should be nonsingular. In reality, however, different variables are often highly correlated, resulting in a near singular $\mathbf{X}'\mathbf{X}$ matrix. This will result in unstable estimates in which a small variation in $\mathbf{e}$, the dependent variable, can have a large impact on $\hat{\beta}$.

RiR addresses this potential problem by introducing a so-called ridge parameter, δ [34]. The introduction of the ridge parameter will yield the following estimator of β:

$$\hat{\beta}_\delta = (\mathbf{X}'\mathbf{X} + \delta I_n)^{-1}(\mathbf{X}'\mathbf{e}),$$

where I_n represents the identity matrix of rank n.

To the best of our knowledge, this technique has not been applied before within the domain of software effort estimation.

3.6 Least Median of Squares Regression (LMS)

LMS is an alternative to robust regression with a breakdown point $\kappa^* = 50\%$ [35]. The breakdown point κ^* is the smallest percentage of incorrect data that can cause an estimator to take on aberrant values [36]. This breakdown point is 0 percent for all the other regression techniques considered in this study, indicating that extreme outliers could have a detrimental influence for these techniques. The LMS will optimize the following objective function:

$$min\ median(\epsilon_i^2),$$

where ϵ_i is the error associated with the ith observation. Although LMS regression is known to be inefficient in some situations [36], this technique has been applied in different domains. However, to the best of our knowledge, the LMS regression has not been applied to the estimation of software effort.

3.7 Multivariate Adaptive Regression Splines (MARS)

MARS is a novel technique introduced by Friedman [37]. MARS is a nonlinear and nonparametric regression technique exhibiting some interesting properties like ease of interpretability, capability of modeling complex nonlinear relationships, and fast model construction. It also excels at capturing interactions between variables and therefore is a promising technique to be applied in the domain of effort prediction. MARS has previously been successfully applied in other domains including credit scoring [38] and biology [39].

MARS fits the data to the following model:

$$e_i = b_0 + \sum_{k=1}^{K} b_k \prod_{l=1}^{L} h_l(x_i(j)),$$

where b_0 and b_k are the intercept and the slope parameter, respectively. $h_l(x_i(j))$ are called hinge functions and are of the form $max(0, x_i(j) - b)$ in which b is called a knot. It is possible to model interaction effects by taking the product of multiple hinge functions. Hence, this model allows for a piecewise linear function by adding multiple hinge functions.

The model is constructed in two stages. In the first stage, called the forward pass, MARS starts from an empty model and constructs a large model by adding hinge functions to overfit the data set. In the second stage, the algorithm removes the hinge functions associated with the smallest increase in terms of the Generalized Cross Validation (GCV) criterion

$$GCV_K = \frac{\sum_{i=1}^{N}(e_i - \hat{e}_{iK})^2}{\left(1 - \frac{C(K)}{N}\right)^2}.$$

Here, $C(K)$ represents a model complexity penalty which is dependent on the number of hinge functions in the model, while the numerator measures the lack of fit of a model with K hinge functions, $\hat{e}_{iK}$. Both LMS and MARS are illustrated on the Desharnais data set in Fig. 1b.

3.8 Classification and Regression Trees (CART)

CART is an algorithm that takes the well-known idea of decision trees for classification [40] and adopts it to

continuous targets. CART constructs a binary tree by recursively splitting the data set until a stopping criterion is met [41]. The splitting criterion used in this study is the least squared deviation:

$$min \sum_{i \in L} (e_i - \overline{\mathbf{e}}_L)^2 + \sum_{i \in R} (e_i - \overline{\mathbf{e}}_R)^2.$$

The data set is split in a left node (L) and a right node (R) in a way that the sum of the squared differences between the observed and the average value is minimal. A minimum of 10 observations at each terminal node is set to halt further tree construction. In retrospect, the fully grown tree is pruned to avoid overfitting on the training set. Figs. 1c and 1d, respectively, present the estimation function and the accompanying binary regression tree for the Desharnais data set.

The good comprehensibility of regression trees can be considered a strong point of this technique. To determine the effort needed for a new project, it is sufficient to select the appropriate branches based on the characteristics of the new project. It is possible to construct an equivalent rule set based on the obtained regression tree.

This technique has previously been applied within a software effort prediction context where it consistently was found to be one of the better performing techniques [14], [15], [33], [42], [43].

3.9 M5

Introduced by Quinlan [44], the model tree technique (M5) can be considered as an extension to CART. A model tree will fit a linear regression to the observations at each leaf instead of assigning a single value like CART.

The model tree algorithm used in this study is the M5′ algorithm which is a variant of the original M5 algorithm [45]. A binary decision tree is induced by recursively applying the following splitting criterion, similarly to CART:

$$min\left(\frac{\mathbf{e}_L}{\mathbf{e}_L + \mathbf{e}_R} \times stdev(\mathbf{e}_L) + \frac{\mathbf{e}_R}{\mathbf{e}_L + \mathbf{e}_R} \times stdev(\mathbf{e}_R)\right).$$

Instead of taking the absolute deviations into account, as is the case with CART, the M5′ algorithm applies a splitting criterion based on standard deviation. After growing and pruning the decision tree, a linear regression is fitted to the observations at each leaf. This regression only considers attributes selected by the different attribute conditions on the nodes, thus resulting in a tree-based piecewise linear model. Finally, a smoothing process is applied to compensate for possible discontinuities that may occur between adjacent linear models at the different leaves.

The use of a model tree algorithm should allow for a more concise representation and higher accuracy compared to CART [44].

3.10 MLP

Neural networks are a nonlinear modeling technique inspired by the functioning of the human brain [46], [47], [48] and have previously been applied in the context of software effort estimation [12], [49], [50]. We further discuss MultiLayered Perceptrons (MLPs) which are the most commonly used type of NNs that are based upon a network of neurons arranged in an input layer, one or more hidden layers, and an output layer in a strictly feedforward manner. Each neuron processes its inputs and generates one output value via a transfer function which is transmitted to the neurons in the subsequent layer. The output of hidden neuron i is computed by processing the weighted inputs and its bias term $b_i^{(1)}$ as follows:

$$h_i = f^{(1)}\left(b_i^{(1)} + \sum_{j=1}^{n} \mathbf{W}_{ij} x_j\right).$$

$\mathbf{W}$ is the weight matrix whereby $\mathbf{W}_{ij}$ denotes the weight connecting input j to hidden unit i. In an analogous way, the output of the output layer is computed as follows:

$$z = f^{(2)}\left(b^{(2)} + \sum_{j=1}^{n_h} \mathbf{v}_j h_j\right),$$

with n_h the number of hidden neurons and $\mathbf{v}$ the weight vector, whereby $\mathbf{v}_j$ represents the weight connecting hidden unit j to the output neuron. The bias term has a similar role as the intercept in regression.

MLP neural networks with one hidden layer are universal approximators, able to approximate any continuous function [51]. Therefore, in this study, an MLP with one hidden layer was implemented. The network weights $\mathbf{W}$ and $\mathbf{v}$ are trained with the algorithm of Levenberg-Marquardt [52]. In the hidden layer, a log sigmoid transfer function has been used, while in the output layer, a linear transfer function is applied. The topology of the neural network is adjusted during training to better reflect specific data sets. In Figs. 1e and 1f, respectively, the MLP estimation function and the accompanying network are given for a subset of the Desharnais data set to increase readability. In the hidden layer, two hidden neurons with log sigmoid transfer functions are used for illustration purposes.

3.11 Radial Basis Function Networks (RBFN)

Radial Basis Function Networks are a special case of artificial neural networks, rooted in the idea of biological receptive fields [53]. An RBFN is a three-layer feedforward network consisting of an input layer, a hidden layer typically containing multiple neurons with radial symmetric gaussian transfer functions, and a linear output layer. Due to the continuous target, a special type of RBFN is used, called Generalized Regression Neural Networks [54]. Within such networks, the hidden layer contains a single neuron for each input sample presented to the algorithm during training. The output of the hidden units is calculated by a radial symmetric gaussian transfer function, $\mathrm{radbas}(x_i)$:

$$\mathrm{radbas}(x_i) = e^{-\|x_k - x_i\| \times b^2},$$

where x_k is the position of the kth observation in the input space, $\|.\|$ the euclidean distance between two points, and b is a bias term. Hence, each kth neuron has its own receptive field in the input domain, a region centered on x_k with size proportional to the bias term, b. The final effort estimates are obtained by multiplying the output of the hidden units with the vector consisting of the targets associated with the

cluster centroids c_k, and then inputting this result into a linear transfer function.

The applicability of RBFN has recently been illustrated within the domain of software effort estimation [55], [56].

3.12 CBR

CBR is a technique that works similarly to the way in which an expert typically estimates software effort; it searches for the most similar cases and the effort is derived based on these retrieved cases. This technique is commonly used in software effort estimation, e.g., [12], [16], [20], [43], [49]. Typically, the euclidean distance with rescaled attributes is used in retrieving the most similar case:

$$\text{Distance}(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{\sum_{k=1}^{n} (x_i(k) - x_j(k))^2}.$$

The rescaling is done by subtracting the minimum and dividing by the difference between maximum and minimum of an attribute. Only relevant attributes should be taken into account when calculating the euclidean distance; in line with [12], [14], only attributes characterized by significant differences in effort are selected as found by applying a t-test in case of binary attributes and an ANOVA test otherwise. In both cases, only attributes with significant differences in effort at $\alpha = 95\%$ are retained during effort estimation. A final issue is the number of analogies to consider. In some studies, it is argued that no significant differences are found when retrieving more than one case, while other studies report a decrease in accuracy if more cases are retrieved [57]. Therefore, multiple alternatives are considered ($\mathrm{k} = 1$, $\mathrm{k} = 2$, $\mathrm{k} = 3$, and $\mathrm{k} = 5$). The final effort is determined by taking the average effort of the retrieved cases.

3.13 Least Squares SVM (LS-SVM)

SVM is a nonlinear machine learning technique based on recent advances in statistical learning theory [58]. SVMs have recently become a popular machine learning technique, suited both for classification and regression. A key characteristic of SVM is the mapping of the input space to a higher dimensional feature space. This mapping allows for an easier construction of linear regression functions. LS-SVM for regression is a variant of SVM in which the goal is to find a linear function $f(\mathbf{x}_i)$ in a higher dimensional feature space minimizing the squared error r_i^2 [59]. The function $f(\mathbf{x}_i)$ takes the following form:

$$f(\mathbf{x}_i) = <\mathbf{w}, \phi(\mathbf{x}_i)> + b,$$

with $\mathbf{w} \in \mathbb{R}^n$ the weight vector in the input space, ϕ a nonlinear function providing the mapping from the input space to a higher (possibly infinite) dimensional feature space, and $b \in \mathbb{R}$ a bias parameter. The function $f(\mathbf{x}_i)$ is determined by solving the following convex optimization problem:

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + \gamma\frac{1}{2}\sum_{i=1}^{N} r_i^2 \\ \text{subject to} \quad & e_i = \mathbf{w}^T\phi(\mathbf{x}_i) + b + r_i, \quad i = 1 \ldots N. \end{aligned}$$

As can be seen from the objective function, a tradeoff is made between the minimization of the squared error, r_i^2, and minimizing the dot product of the weight vectors, $\mathbf{w}^T\mathbf{w}$, by an optimization parameter γ. The Lagrangian of the problem takes the following form:

$$\frac{1}{2}\mathbf{w}^T\mathbf{w} + \gamma\frac{1}{2}\sum_{i=1}^{N} r_i^2 - \sum_{i=1}^{N} \alpha_i\{\mathbf{w}^T\phi(\mathbf{x}_i) + b + r_i - e_i\},$$

where $\alpha_i \in \mathbb{R}$ are the Lagrange multipliers. The problem is reformulated in its dual form giving way to the following equation:

$$e_i = \sum_{i=1}^{N} \alpha_i < \phi(\mathbf{x}), \phi(\mathbf{x}_i) > + b.$$

At this point, the kernel function is applied which will compute the dot product in the higher dimensional feature space by using the original attribute set. In this study, a Radial Basis Function kernel was used since it was previously found to be a good choice in case of LS-SVMs [60]:

$$K(\mathbf{x}, \mathbf{x}_i) = e^{-\frac{\|\mathbf{x}-\mathbf{x}_i\|^2}{2\sigma^2}},$$

where σ is a kernel parameter determining the bandwidth of the kernel. It has been shown that the bandwidth is an important parameter of the generalization behavior of a kernel method [61].

SVMs are a popular technique which has been applied in various domains. Since this is a rather recent machine learning technique, its suitability in the domain of software effort estimation has only been studied to a limited extent [62].

4 Empirical Setup

4.1 Data Sets

Nine data sets from companies of different industrial sectors are used to assess the techniques discussed in Section 3. While other software effort estimation data sets exist in the public domain (e.g., a study of Mair et al. identified 31 such data sets [63]), the majority of these data sets are rather small. The overview of Mair et al. contained only three data sets pertaining to over 50 projects, the Coc81, CSC, and Desharnais data sets. The CSC data set, however, focuses on differences between estimation approaches instead of project characteristics and is therefore not included in this study. Investigating recent literature, three other data sets in the public domain were identified, the Cocnasa, Maxwell, and USP05 data sets. Furthermore, researchers having access to data sets pertaining to over 150 projects were contacted, as well as several companies involved in effort estimation. As such, access to four additional software effort estimation data sets was obtained (the Experience, ESA, ISBSG, and Euroclear data sets).

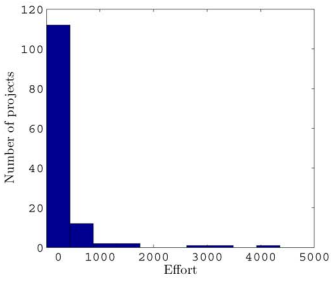
The data sets typically contain a unique set of attributes that can be categorized as follows:

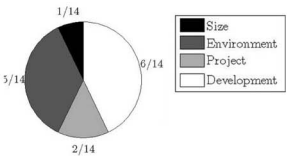
- *Size attributes* are attributes that contain information concerning the size of the software project. This information can be provided as Lines Of Code

TABLE 3
Overview Software Effort Prediction Data Sets

ESA
Skewness: 4.89
Kurtosis: 30.13
Minimum effort: 3 *Minimum Kloc: 1.5*
Mean effort: 264 *Mean Kloc: 56*
Max effort: 4361 *Max Kloc: 413*

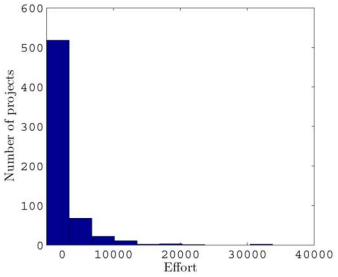
Number of observations: 131
Number of attributes: 14
Previously used in other studies, e.g. [15], [64], [65]
Reference: The ESA initiative for Software Productivity Benchmarking and Effort Estimation
http://www.esa.int/esapub/bulletin/bullet87/greves87.htm

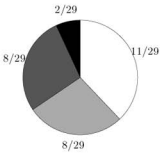




Experience
Skewness: 4.45
Kurtosis: 32.30
Minimum effort: 55 *Minimum FP: 7.14*
Mean effort: 4248 *Mean FP: 728*
Max effort: 67576 *Max FP: 14092*

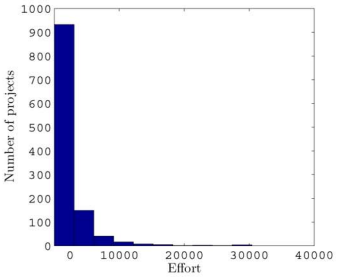
Number of observations: 627
Number of attributes: 29
Previously used in other studies, e.g. [14], [50], [64], [66], [67]
Reference: http://www.fisma.fi

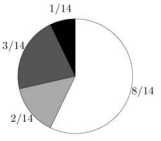




ISBSG
Skewness: 4.40
Kurtosis: 30.50
Minimum effort: 16 *Minimum FP: 4*
Mean effort: 4226 *Mean FP: 374*
Max effort: 60826 *Max FP: 7400*

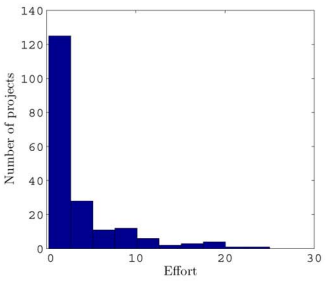
Number of observations: 1160
Number of attributes: 14
Previously used in other studies, e.g. [13], [16], [68]
Reference: http://www.isbsg.org





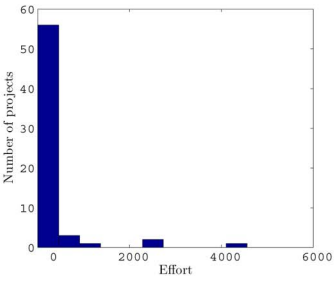
USP05
Skewness: 2.10
Kurtosis: 7.14
Minimum effort: 0.5 *Minimum FP: 0*
Mean effort: 7.25 *Mean FP: 25*
Max effort: 50 *Max FP: 321*

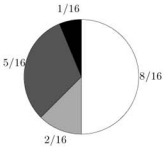
Number of observations: 193
Number of attributes: 14
Previously used in other studies, e.g. [16], [69]
Reference: [16], [69]
http://www.promisedata.org/?p=48



Coc81
Skewness: 4.37
Kurtosis: 23.08
Minimum effort: 5.9 *Minimum Kloc: 1.98*
Mean effort: 683 *Mean Kloc: 77*
Max effort: 11400 *Max Kloc: 1150*

Number of observations: 63
Number of attributes: 16
Previously used in other studies, e.g. [13], [55], [70]–[72]
Reference: [7]
http://www.promisedata.org/?p=6





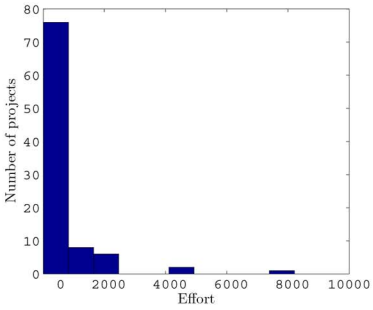
(LOC), Function Points, or some other measure. Size related variables are often considered to be important attributes to estimate effort [7].

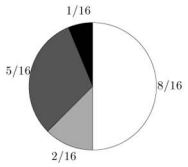
- *Environment information* contains background information regarding the development team, the company, the project itself (e.g., the number of developers involved and their experience), and the sector of the developing company.
- *Project data* consist of attributes that relate to the specific purpose of the project and the project type. Also attributes concerning specific project requirements are placed in this category.

TABLE 3 (Continued)

Cocnasa
Skewness: 4.19
Kurtosis: 24.81
Minimum effort: 8.4 *Minimum Kloc: 0.9*
Mean effort: 624 *Mean Kloc: 94*
Max effort: 8211 *Max Kloc: 980*

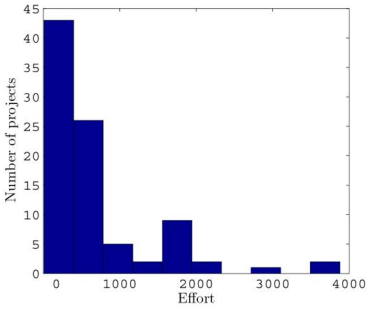
Number of observations: 93
Number of attributes: 16
Previously used in other studies, e.g. [70], [72]
Reference: http://www.promisedata.org/?p=35





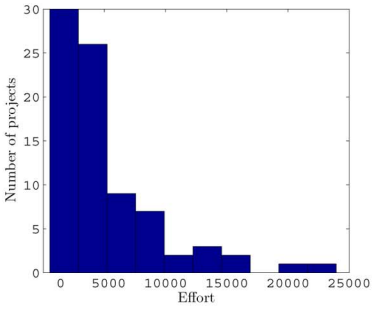
Euroclear
Skewness: 2.25
Kurtosis: 8.49
Minimum effort: 24
Mean effort: 1402
Max effort: 7758

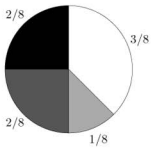
Number of observations: 90
Number of attributes: 11
Data set has not been used in previous studies
Reference: http://www.euroclear.com



Desharnais
Skewness: 1.97
Kurtosis: 7.36
Minimum effort: 546 *Minimum FP: 62*
Mean effort: 5046 *Mean FP: 287*
Max effort: 23940 *Max FP: 1116*

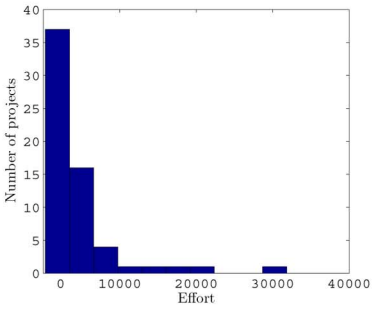
Number of observations: 81
Number of attributes: 9
Previously used in other studies, e.g. [20], [49], [64], [73]–[75]
Reference: [76]
http://www.promisedata.org/?p=9

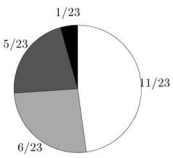




Maxwell
Skewness: 3.27
Kurtosis: 15.52
Minimum effort: 583 *Minimum FP: 48*
Mean effort: 8223 *Mean FP: 673*
Max effort: 63694 *Max FP: 3643*

Number of observations: 62
Number of attributes: 23
Previously used in other studies, e.g. [13], [74]
Reference: [77]
http://www.promisedata.org/?p=108





- *Development related variables* contain information about managerial aspects and/or technical aspects of the developed software projects, such as the programming language or type of database system that was used during development.

Table 3 provides an overview of the data sets, including number of attributes, observations, and previous use. The skewness and kurtosis as well as the minimum, mean and maximum of effort and size in Klocs or FP is given. A histogram of effort is provided for each data set in the center while, on the right-hand side, the partitioning of the different attributes across the four attribute types is shown. From this overview, the inherent difficulties to construct software effort estimation models become apparent. Data sets typically are strongly positively skewed indicating many "small" projects and a limited number of "large" outliers. Also, data sets within this domain are typically small as compared to other domains. Most data mining techniques benefit from having more observations to learn from. Table 4 further details a number of basic characteristics of the data sets including whether the data were collected from a single or multiple companies, the application domain of the software, the size measure used, and the years during which the information was collected.

TABLE 4
Characteristics of Software Effort Estimation Data Sets

Data set	Single/multi company	Application domains	Size measure	Range years
ESA	M	Space/military	Kloc	1983-1996
Experience	M	Manufacturing, banking, retail,...	FP [2]	1987-2010
ISBSG	M	Accounting, banking, trade,...	FP	1989-2009
USP05	S	Student projects	FP	2005
Coc81	S	Engineering, science, finance,...	Kloc	1970-1981
Cocnasa	M	Space/military	Kloc	1981-1999
Euroclear	S	Finance	-	2006-2008
Desharnais	S	Unknown	FP	1981-1988
Maxwell	S	Finance	FP	1985-1993

[2] *A five point scale is used to measure complexity weights instead of a three point scale.*

4.2 Data Preprocessing

The first important step in each data mining exercise is data preprocessing. In order to correctly assess the techniques discussed in Section 3, the same data preprocessing steps are applied to each of the nine data sets.

First, starting from the raw data set, the data used to learn and validate the models are selected; only attributes that are known at the moment when the effort is estimated are taken into account (e.g., duration or cost are not known and therefore not included in the data set). An implicit assumption made in most software effort estimation studies is that size-related attributes are taken for granted. However, in reality such attributes are often a result of an estimation process on their own. This remark is also echoed, e.g., by Jørgensen, stating that "a program module's size and degree of complexity...are typically based on expert judgment" [24]. However, this assumption is made (but rarely mentioned) not only in this study, but also in almost all studies in the domain of software effort estimation. Furthermore, some of the data sets include an indication about the reliability of observations. Taking this information into account, the observations with a higher possibility of being incorrect are discarded. In Table 3, an overview of the number of retained attributes and observations is provided.

Second, since some of the techniques are unable to cope with missing data (e.g., OLS regression), an attribute is removed if more than 25 percent of the attribute values are missing. Otherwise, for continuous attributes, median imputation is applied in line with [78]. In case of categorical attributes, a missing value flag is created if more than 15 percent of the values are missing; else, the observations associated with the missing value are removed from the data set. Since missing values often occur in the same observations, the number of discarded projects turned out to be low. In the Appendix, the data preprocessing is illustrated for the ISBSG data set as this is the largest data set, both in number of attributes and number of projects.

Finally, coarse classification with k-means clustering is applied in case of categorical attributes with more than eight different categories (excluding the missing value flag). Afterward, the categorical variables are transformed into binary variables using dummy encoding. No other preprocessing steps are performed on the data.

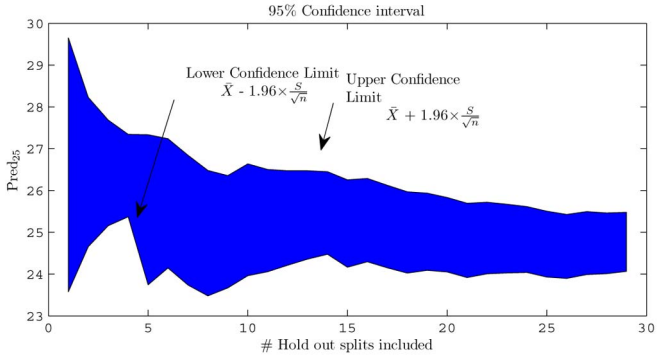


Fig. 2. Example of 95 percent confidence interval.

Data mining techniques typically perform better if a larger training set is available. On the other hand, a part of the data needs to be put aside as an independent test set in order to provide a realistic assessment of the performance. As can be seen from Table 3, the smallest data set contains 62 observations, while the largest contains up to 1,160 observations. In case of data sets containing more than 100 observations, hold out splitting is applied; otherwise, leave one out cross validation (LOOCV) is used.

In case of holdout splitting, the initial data set is randomly partitioned into two disjoint sets, i.e., a training and test set consisting of, respectively, 2/3 and 1/3 of the observations. The model is induced on the training set while the independent test set is used to evaluate the performance. To account for a potential bias induced by the holdout split, this procedure is repeated 20 times. It is argued by Kirsopp and Shepperd [79] that "ideally more than 20 sets should be deployed." With repeatable sampling, a confidence interval indicating the reliability of the performance estimate can be constructed. After empirical investigation and in line with [79], 20 random holdout samples were taken in case of data sets containing more than 100 observations. Further sampling typically did not yield a significant reduction in variation, as can be seen from Fig. 2.

In case of LOOCV, iteratively one observation is selected as the test set while the remaining observations are used as the training set. The total error is found by summing the errors on the test set (or taking the average of the errors depending on the evaluation metric) in each step. The LOOCV approach is computationally more expensive since as many models need to be estimated as there are observations in the data set, but guarantees that, as much as possible, observations are used to learn from [80]. This approach has previously been adopted in the field of software effort prediction, see, e.g., [20], [64], [81]. Note that there still is a discussion on whether k-fold cross validation or LOOCV is best; however, Myrtveit et al. note that LOOCV is more in line with real world situations [81].

4.3 Technique Setup

Several of the estimation techniques discussed in Section 3 have adjustable parameters, also referred to as hyperparameters, which enable a model to be adapted to a specific problem. When appropriate, default values are used based on previous empirical studies and evaluations reported in

the literature. If no generally accepted default parameter values exist, then these parameters are tuned using a grid-search procedure. In other words, a set of candidate parameter values is defined and all possible combinations are evaluated by means of a split-sample setup. The models are induced on 2/3 of the training data and the remainder is used as a validation set. The performance of the models for a range of parameter values is assessed using this validation set. The parameter values resulting in the best performance are selected and a final model is trained on the full training set. The MdMRE performance measure (see Section 4.5) was selected for hyperparameter tuning since this measure is a generally accepted metric for software effort estimation as it is outlier resistant.

4.4 Input Selection

A second factor impacting the performance of software effort prediction models is input selection. Typically, similar, or occasionally better performance can be obtained by inducing models from a data set containing less, but highly relevant attributes, yielding a more concise and comprehensible model [82]. Therefore, a generic input selection procedure is applied in which a subset of highly predictive attributes is selected, discarding irrelevant variables.

A wrapper approach is adopted which evaluates candidate attribute subsets by executing a selected learning algorithm on a reduced data set. Starting with the original data set, an iterative backward input selection procedure is adopted in which, in each step, as many models are induced as there are variables left. Each of these models include all the remaining variables except one. The performance of the estimated models is compared, and the best performing attribute subset is selected. This procedure is repeated until only one attribute remains in the data set. This approach is computationally expensive since, in each iteration, as many models as remaining attributes need to be estimated. The performance of the sequentially best models with a decreasing number of attributes is plotted, see Fig. 4. In the beginning, the performance typically remains stable or even increases while discarding attributes [72]. When the size of the attribute set drops below a certain number of attributes, the performance of the model drops sharply. The model at the elbow point is considered to incorporate the optimal trade-off between maximizing the performance and minimizing the number of attributes. The performance at this elbow point is reported in Section 5.

Algorithm 1 provides a formal description of the followed procedure in case of data sets containing more than 100 observations. Otherwise, a cross validation-based alternative is adopted.

Algorithm 1. Pseudocode of backward input selection

1: Let $\mathcal{D}^n_{tr,l}$ and $\mathcal{D}^n_{te,l}$ be the l^{th} ($l = 1 \ldots 20$) random holdout split of a data set with n attributes and N observations
2: **for** $j = n$ **to** 1 **do**
3: **for** $k = 1$ **to** j **do**
4: Exclude attribute k from data sets $\mathcal{D}^j_{tr,l}$ and $\mathcal{D}^j_{te,l}$
5: **for** $l = 1$ **to** 20 **do**
6: Induce model from $\mathcal{D}^j_{tr,l}$
7: Calculate model performance $P^j_{k,l}$ on $\mathcal{D}^j_{te,l}$
8: **end for**
9: Calculate mean performance over all holdout splits: $P^j_k = \frac{1}{20}\sum_{l=1}^{20} P^j_{k,l}$
10: **end for**
11: Remove attribute $\mathbf{x}'(m)$ from $\mathcal{D}^j$ where $P^j_m = \max_k(P^j_k)$ resulting in $\mathcal{D}^{j-1}$
12: **end for**
13: Plot(j, P^j_m) with $j = 1, \ldots, n$
14: Select elbow point with optimal tradeoff between performance and number of variables

4.5 Evaluation Criteria

A key question to any estimation method is whether the predictions are accurate; the difference between the actual effort, e_i, and the predicted effort, $\hat{e}_i$, should be as small as possible. Large deviations between e_i and $\hat{e}_i$ will have a significant impact on the costs related to the development of software. A criterion often used in the literature on cost estimation models is the Magnitude of Relative Error (MRE) [83]. The MRE is calculated for each observation and is defined as

$$MRE_i = \frac{|e_i - \hat{e}_i|}{e_i}.$$

Based on the MRE criterion, a number of accuracy measures are defined. The MRE value of individual predictions can be averaged, resulting in the Mean MRE (MMRE):

$$MMRE = \frac{100}{N}\sum_{i=1}^{N}\frac{|e_i - \hat{e}_i|}{e_i}.$$

Although it is a commonly used measure (see also Table 2), the MMRE can be highly affected by outliers [84]. To address this shortcoming, the MdMRE metric has been proposed which is the median of all MREs. This metric can be considered more robust to outliers, and is therefore preferred over the MMRE:

$$MdMRE = 100 \times median(MRE).$$

A complementary accuracy measure is Pred_L, the fraction of observations for which the predicted effort, $\hat{e}_i$, falls within L percent of the actual effort, e_i:

$$Pred_L = \frac{100}{N}\sum_{i=1}^{N}\begin{cases} 1, & \text{if } MRE_i \leq \frac{L}{100}, \\ 0, & \text{otherwise.} \end{cases}$$

Typically, the Pred_{25} measure is considered, looking at the percentage of predictions that are within 25 percent of the actual values.

The Pred_{25} can take a value between 0 and 100 percent, while the MdMRE can take any positive value. It is often difficult to compare results across different studies due to differences in empirical setup and data preprocessing, but a typical Pred_{25} lies in the range of 10 to 60 percent, while the MdMRE typically attains values between 30 and 100 percent.

Besides Pred_{25} and MdMRE, we also compared the techniques using a correlation metric. As the data are not normally distributed (see also Table 3), a rank correlation measure is adopted, which is a measure of the monotonic relationship between e_i and $\hat{e}_i$. More specifically, the Spearman's rank correlation coefficient, r_s, is used since

this nonparametric correlation coefficient does not assume a normal distribution of the underlying data [28]. The Spearman's rank correlation takes a value between -1 and $+1$ with $+1$ (-1) indicating a perfect positive (negative) monotonic relationship between the actual values and the predicted values. The Spearman's rank correlation is defined as

$$r_s = 1 - \frac{6\sum_{i=1}^{N} d_i^2}{N(N^2-1)},$$

whereby d_i represents the difference between the ordinal ranks assigned to each of the observations. In case of equal ranks, the average rank is assigned.

Please note that there still is no consensus of opinion as to which metric is the most suited within the field of software effort prediction [81]. Earlier studies pointed out that MRE-based metrics, and especially the MMRE, sometimes prefer models that underestimate to a model that estimates the correct value [85]. A number of other metrics have been proposed in recent literature, such as MER-based metrics [86], coefficient of determination (R^2), mean absolute residual, standard deviation of errors, as well as a number of alternatives like the logarithmic standard deviation [81], [87]. None of these other metrics, however, have gained wide acceptance because they suffer from a flaw or limitation. For example, Foss et al. note that both the standard deviation as well as the logarithmic standard deviation make certain assumptions as to whether the data is homo or heteroscedastic [87]. R^2 and the mean absolute residual are, on the other hand, known to be outlier sensitive [21]. While the rank reversal problem (a better performing model mistakenly found to be less accurate) cannot be ruled out, we believe our selection of metrics (MdMRE, Pred_{25}, and r_s) and the empirical setup of the study to be robust in this respect.

4.6 Statistical Tests

A procedure described in Demšar [88] is followed to statistically test the results of the benchmarking experiment. In the first step of this procedure the Friedman test [89] is performed, which is a nonparametric equivalent of the well-known ANalysis Of Variance (ANOVA) test. The null hypothesis of the Friedman test states that all techniques perform equivalent. The test statistic is defined as

$$\chi_F^2 = \frac{12P}{k(k+1)}\left[\sum_{j=1}^{k} R_j^2 - \frac{k(k+1)^2}{4}\right],$$

with R_j the average rank of algorithm $j = 1, 2 \ldots k$ over P data sets. Under the null hypothesis, the Friedman test statistic is distributed according to χ_F^2 with $k-1$ degrees of freedom, at least when P and k are big enough ($P > 10$ and $k > 5$). Otherwise, exact critical values are used based on an adjusted Fisher z-distribution. When comparing the results with input selection to the results without input selection, k equals 2 and P equals 9.

If the null hypothesis of equivalent performing techniques is rejected by the Friedman test, a posthoc Bonferroni-Dunn test [90] is applied to compare the models. The posthoc Bonferroni-Dunn test is a nonparametric alternative of the Tukey test and is defined as

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6P}},$$

with critical value q_α based on the Studentized range statistic divided by $\sqrt{2}$, and an additional Bonferroni correction by dividing the confidence level α by the number of comparisons made, $(k-1)$, to control for family wise testing. This results in a lower confidence level and thus in higher power. The difference in performance of the best performing technique and other techniques is significant if the corresponding average ranks (AR) differ by at least the Critical Distance (CD).

The previous tests are performed in order to compare the results across the different data sets. Additionally, to compare the performance of two models on a single data set, the nonparametric Wilcoxon Matched Pairs test [91] (in case of the MdMRE) and the parametric t-test [28] (in case of Pred_{25} and correlation) are performed. The Wilcoxon Matched Pairs test compares the ranks for the positive and negative differences in performance of two models, and is defined as

$$min\left(\sum_{d_i>0} R(d_i) + \frac{1}{2}\sum_{d_i=0} R(d_i), \sum_{d_i<0} R(d_i) + \frac{1}{2}\sum_{d_i=0} R(d_i)\right),$$

with $R(d_i)$ the rank of the difference in performance between two models, ignoring signs. This test statistic follows approximately a standard normal distribution. The t-test is a general statistical test which is typically used to assess the difference between two responses. Under the null hypothesis, this test statistic follows a Student t-distribution.

5 Results

This section reports on the results of the techniques discussed in Section 3. The results both with and without application of the backward input selection procedure, as explained in Section 4.4, are provided in Tables 5, 6, and 7, respectively, for the MdMRE, Pred_{25}, and Spearman's rank correlation. The top panels show the results without backward input selection and the bottom panels with backward input selection. The last column of each table displays the Average Ranks for the different techniques. The techniques are ranked according to their performance on each data set, rank 1 indicating the best performance and rank 16 the worst. The AR are then calculated by averaging the ranks across the different data sets.

The best performing technique is reported in bold and underlined. Results that are not significantly different from the best performing technique at 95 percent are tabulated in boldface font, while results significantly different at 99 percent are displayed in italic script. Results significant at the 95 percent level but not at the 99 percent level are displayed in normal script.

5.1 Techniques

The results of the different modeling techniques are compared by first applying a Friedman test, followed by a

TABLE 5
Test Set MdMRE Performance

Technique/Data set	ISBSG	Experience ESA		USP05	Euroclear	Cocnasa	Coc81	Desharnais	Maxwell	AR	
RiR	70.4	57	59.9	73.7	69	87.7	336	36.9	41.9	11.1	Without backward input selection
M5	67	47.4	56.1	73.5	64.4	61.1	107	32.4	57.1	9.67	
LMS	62.3	45	52.2	47.8	54.4	36.8	66.4	32.8	44.6	5.11	
OLS	71.6	54.5	60.8	91.4	64.9	127	483	31	61.6	11.7	
BC + OLS	51	49.2	49.3	32.8	54.1	170	80.2	33.2	50.8	5.56	
Log + OLS	38.7	37.6	46.2	33.7	60.8	30.4	27.6	27.5	36.9	2.44	
CART	55.7	42.8	54.7	42.8	56.5	47	76.4	33.4	48.4	5.89	
MARS	59.6	54.2	62.3	74.8	66.9	46.6	66.6	29.8	51.9	8	
RoR	318	108	139	479	175	700	69.7	52.7	68.8	14.4	
LS-SVM	60.5	62.2	50.5	43.2	68.8	35.4	105	35.6	45.7	8.22	
MLP	115	98.4	80.4	72.1	61	80.6	99	36.1	65.5	12.4	
CBR k = 1	58	62.8	56.6	28.7	47.4	43.2	75.4	47.2	53.4	7.22	
CBR k = 2	52.7	59.1	53	31.5	52.7	41.7	86.1	45.8	61.8	6.89	
CBR k = 3	53.6	60.3	52	32.5	54.6	49.2	85.6	45.7	55.4	7.11	
CBR k = 5	57.3	61.9	53.9	33.3	60.1	58.3	91.3	40	56	8.78	
RBFN	61.6	57.6	70.2	47.7	56.6	85.6	89.2	100	100	11.4	
RiR	56.6	45.6	55.4	45.9	58	44.3	95.3	33.6	32.3	10.8	With backward input selection
M5	46.2	41.7	54.4	53.5	58.1	48.6	82.8	29.4	46.6	10.8	
LMS	50.7	38.1	50.3	36.6	42.6	28.1	47.6	27.9	38.4	4.33	
OLS	58.5	48.8	58.3	51.2	64.4	51.3	177	29.4	48.2	13.9	
BC + OLS	47.2	47.2	48.8	28.5	44.8	44.7	62.9	31.1	39.8	6.56	
Log + OLS	34.7	36.2	45.8	28.8	56	25.2	30.5	27.5	37.6	2.89	
CART	56.4	41	53.3	38.9	51.2	45	65.3	30.4	45.5	9.33	
MARS	49.9	39.6	60.2	61.5	53.8	39.3	65	26.8	41	8.67	
RoR	100	40.7	51	39.7	44.6	34.9	64.9	31	44.9	7.67	
LS-SVM	39.7	41.8	52.1	38.8	63.2	42.6	68.1	33.9	41	8.78	
MLP	56.7	44.8	57.1	48.1	51.7	38.5	79	25.4	44.1	9.89	
CBR k = 1	45.6	51	55.7	25.8	47.4	28.2	65.4	43.4	43.2	8.22	
CBR k = 2	45.6	47.6	51.2	34.8	46.1	28.3	75.7	37.1	39.1	7.56	
CBR k = 3	46.4	45.2	50	33.1	47	37.5	82.5	36.7	38.1	7.33	
CBR k = 5	46.9	42.6	48.4	31.8	48	44	74.8	34.6	36.2	6.78	
RBFN	51.6	54.2	59.5	32.6	54.6	82.5	88.6	47.9	40.6	12.4	

TABLE 6
Test Set $\mathrm{Pred}_2 5$ Performance

Technique/Data set	ISBSG	Experience ESA		USP05	Euroclear	Cocnasa	Coc81	Desharnais	Maxwell	AR	
RiR	21.9	22.9	22.6	21.7	16.7	15.1	9.52	39.5	32.3	10.1	Without backward input selection
M5	26.1	27.5	24.5	20.7	17.8	24.7	12.7	38.3	14.5	8.33	
LMS	21	29.2	25.7	31.1	23.3	40.9	23.8	40.7	17.7	5.67	
OLS	23.9	24.4	22.7	17.8	17.8	15.1	6.35	45.7	25.8	9.89	
BC + OLS	25.2	24.5	25.4	42.2	23.3	7.53	14.3	43.2	24.2	6.17	
Log + OLS	33.1	33.8	27	38.9	20	45.2	49.2	43.2	25.8	2.94	
CART	22.7	30.6	23.5	35.3	21.1	22.6	11.1	34.6	30.6	7.33	
MARS	24.5	26	22.2	18.5	20	35.5	19	38.3	27.4	7.22	
RoR	9.43	15.4	8.97	6.92	12.2	3.23	20.6	25.9	16.1	14	
LS-SVM	24.7	21.5	26.5	32.8	16.7	38.7	12.7	35.8	25.8	7.78	
MLP	13.9	14.7	16.8	20.3	21.1	15.1	15.9	35.8	19.4	11.4	
CBR k = 1	18.8	19.5	23.4	48.8	18.9	33.3	15.9	23.5	22.6	9.28	
CBR k = 2	22	20.9	23.6	47.8	28.9	33.3	19	27.2	17.7	7.5	
CBR k = 3	25.8	21.6	24.4	42.2	18.9	26.9	11.1	33.3	22.6	8.22	
CBR k = 5	25.5	20	23.2	43.5	18.9	19.4	11.1	35.8	30.6	8.17	
RBFN	16.3	24	18.2	35.8	20	15.1	9.52	2.47	14.5	12.1	
RiR	22.7	28.8	24.8	27.4	17.8	35.5	7.94	38.3	35.5	10.4	With backward input selection
M5	28.1	31.6	25.3	26.1	20	30.1	12.7	38.3	29	8.94	
LMS	24.9	34.5	23.9	38	32.2	44.1	27	42	30.6	5.94	
OLS	23.4	27.2	24	21.1	17.8	31.2	6.35	38.3	28.6	12.4	
BC + OLS	25.5	24.1	24.1	46.5	21.1	29	19	42	30.4	8.5	
Log + OLS	36.3	34.6	26.7	45.5	20	49.5	39.7	43.2	32.1	3.33	
CART	22.8	32.2	24.3	36.2	25.6	28	12.7	35.8	30.4	9.72	
MARS	28.3	34.9	21.2	19.5	22.2	34.4	17.5	45.7	33.9	7.11	
RoR	19.1	30	23.4	29.8	18.9	37.6	15.9	44.4	23.2	10.5	
LS-SVM	32.8	33.3	24.9	40.2	17.8	30.1	17.5	44.4	26.8	7.67	
MLP	23.6	30.5	23.4	30.2	23.3	37.6	17.5	49.4	23.2	8.44	
CBR k = 1	25.9	24	24.7	49.8	18.9	47.3	19	33.3	21.4	8.83	
CBR k = 2	27.2	27.7	25.3	45.1	28.9	45.2	14.3	30.9	39.3	6.06	
CBR k = 3	26.4	26.7	24.8	40.6	22.2	36.6	12.7	33.3	35.7	8.17	
CBR k = 5	22.7	30.6	25.7	44.8	31.1	31.2	9.52	37	32.1	7.89	
RBFN	24.3	25.7	21.2	43.8	23.3	15.1	9.52	28.4	28.6	12	

Bonferroni-Dunn test, as explained in Section 4.6. The Friedman test resulted in a p-value close to zero (p-values between 0.0000 and 0.0002) indicating the existence of significant differences across the applied techniques in all three cases (MdMRE, Pred_{25}, and r_s). In a next step, the Bonferroni-Dunn test to compare the performance of all the models with the best performing model is applied. The results are plotted in Fig. 3. The horizontal axis in these figures corresponds to the average rank of a technique across the different data sets. The techniques are represented by a

TABLE 7
Test Set Spearman's Rank Correlation Performance

	Technique/Data set	ISBSG	Experience	ESA	USP05	Euroclear	Cocnasa	Coc81	Desharnais	Maxwell	AR
Without backward input selection	RiR	**0.807**	**0.743**	**0.736**	**0.808**	**0.492**	*0.681*	*0.706*	**0.81**	**0.746**	**5.78**
	M5	**0.785**	**0.789**	**0.729**	**0.787**	**0.459**	*0.738*	**0.826**	**0.712**	**0.699**	**5.89**
	LMS	0.749	**0.797**	**0.739**	*0.688*	0.434	0.786	*0.765*	**0.78**	0.603	**6.33**
	OLS	**0.775**	**0.747**	**0.731**	**0.779**	0.447	*0.749*	*0.736*	**0.829**	**0.728**	**6**
	BC + OLS	0.738	**0.729**	**0.742**	**0.845**	**0.62**	0.764	*0.638*	**0.82**	**0.669**	**5.33**
	Log + OLS	**0.865**	**0.837**	**0.788**	**0.851**	*0.0375*	**0.889**	**0.963**	**0.863**	**0.793**	**2.44**
	CART	0.732	**0.799**	**0.72**	**0.769**	0.417	*0.699*	*0.674*	0.614	**0.677**	**8.33**
	MARS	*0.72*	**0.759**	**0.717**	**0.736**	*0.41*	**0.806**	*0.695*	*0.497*	0.592	9.11
	RoR	0.747	**0.79**	*0.614*	*0.369*	*0.243*	*0.561*	**0.827**	*0.132*	*0.346*	*10.8*
	LS-SVM	*0.679*	**0.703**	**0.765**	**0.822**	*0.0118*	**0.814**	0.8	**0.736**	**0.761**	**6.33**
	MLP	*0.484*	*0.472*	*0.508*	*0.661*	*0.187*	*0.588*	*0.494*	**0.752**	**0.651**	*12.7*
	CBR k = 1	*0.618*	*0.462*	*0.611*	**0.803**	**0.569**	*0.741*	*0.75*	*0.443*	*0.496*	*10.3*
	CBR k = 2	*0.651*	*0.516*	*0.649*	**0.823**	**0.487**	*0.706*	*0.567*	*0.527*	*0.49*	*10.4*
	CBR k = 3	*0.66*	*0.528*	0.662	**0.831**	0.449	*0.714*	*0.51*	*0.546*	*0.556*	9.89
	CBR k = 5	*0.666*	*0.537*	0.659	**0.807**	0.453	*0.686*	*0.495*	0.597	0.586	*10.3*
	RBFN	*0.26*	*0.0861*	*0.238*	*0.338*	*-0.055*	*0.327*	*0.233*	*-0.315*	*-0.147*	*16*
With backward input selection	RiR	*0.759*	**0.809**	*0.724*	**0.806**	**0.53**	**0.809**	*0.324*	*0.694*	**0.782**	**7.22**
	M5	0.787	**0.821**	*0.731*	**0.826**	**0.532**	**0.847**	**0.771**	*0.716*	**0.749**	**4.17**
	LMS	**0.807**	**0.809**	*0.687*	*0.728*	**0.592**	**0.817**	**0.887**	**0.807**	**0.782**	**5.56**
	OLS	*0.785*	**0.771**	*0.728*	*0.705*	**0.475**	*0.724*	*0.18*	*0.716*	*0.719*	9.72
	BC + OLS	*0.783*	**0.75**	*0.736*	**0.861**	**0.485**	0.805	*0.664*	**0.802**	**0.763**	**6.56**
	Log + OLS	**0.849**	**0.833**	**0.782**	**0.855**	*-0.0839*	**0.91**	**0.947**	**0.864**	**0.809**	**2.78**
	CART	*0.745*	**0.805**	*0.71*	0.794	0.385	*0.76*	*0.702*	*0.588*	*0.624*	9.94
	MARS	**0.802**	**0.832**	*0.697*	0.794	*0.335*	0.797	*0.672*	*0.692*	*0.722*	**8.06**
	RoR	*0.773*	**0.81**	*0.718*	*0.606*	**0.523**	**0.825**	**0.827**	**0.821**	*0.649*	**7**
	LS-SVM	**0.815**	**0.82**	*0.74*	**0.824**	**0.433**	*0.704*	**0.791**	*0.37*	*0.598*	**7.89**
	MLP	*0.664*	**0.758**	*0.696*	*0.713*	*0.085*	*0.737*	*0.476*	**0.765**	*0.595*	*12.2*
	CBR k = 1	*0.735*	*0.663*	*0.645*	*0.763*	**0.569**	**0.833**	*0.66*	*0.526*	*0.616*	*10.7*
	CBR k = 2	*0.713*	0.691	*0.669*	*0.773*	**0.428**	0.8	*0.695*	*0.548*	*0.669*	*11.6*
	CBR k = 3	*0.715*	**0.71**	*0.692*	*0.78*	**0.56**	**0.821**	*0.711*	*0.654*	*0.672*	**9.11**
	CBR k = 5	*0.74*	**0.782**	*0.724*	**0.806**	**0.516**	**0.831**	0.726	*0.683*	*0.615*	**8**
	RBFN	*0.584*	*0.177*	*0.627*	*0.627*	*-0.00552*	*0.324*	*0.228*	*0.376*	*0.554*	*15.6*

horizontal line; the more this line is situated to the left, the better performing a technique is. The left end of this line depicts the average ranking, while the length of the line corresponds to the critical distance for a difference between any technique and the best performing technique to be significant at the 99 percent confidence level. In case of 16 techniques and 9 data sets, this critical distance is 7.0829. The dotted, dashed and full vertical lines in the figures indicate the critical difference at, respectively, the 90, 95, and 99 percent confidence level. A technique is significantly outperformed by the best technique if it is located at the right side of the vertical line.

Data sets in the domain of software effort estimation have specific characteristics [92]. They often have a limited number of observations, are affected by multicollinearity, and are known to be positively skewed and to contain outliers. Different techniques (both linear and nonlinear models, tree/rule induction techniques and case-based reasoning) have been applied in this study that cope with these characteristics in different ways.

It can be seen from Tables 5, 6, and 7 that ordinary least squares regression with logarithmic transformation (Log + OLS) is the overall best performing technique. However, a number of other techniques including least median squares regression, ordinary least squares regression with Box Cox transformation, and CART, are not significantly outperformed by Log + OLS, see Fig. 3. There are a few notable exceptions, such as the Euroclear data set (all three performance measures), the USP05 data set (in case of MdMRE and Pred_{25}), and both the Desharnais as well as the Maxwell data set (only for Pred_{25}). The good performance of Log + OLS can be attributed to the fact that such a transformation typically results in a distribution which better resembles a normal distribution. The range of possible values is also reduced thus limiting the number of outliers. Applying a Jarque-Bera test for normality on the log transformed data, it was found that in all but three cases (USP05, Euroclear, and ISBSG), the null hypothesis of normality could not be rejected at $\alpha = 5\%$. Related to the normality of the distribution is the number of extreme values or outliers. For instance, if an outlier is defined as an observation at a distance of more than 1.5 times the interquartile range of either the first or the third quartile, applying the logarithmic transformation removes all outliers in case of both the Cocnasa and the Coc81 data sets. In case of the ISBSG, Experience, and ESA data sets, less than 2 percent of the observations can be regarded as outliers after applying a log transformation. In case of the USP05 data set, such transformation was less able to reduce the number of outliers as still 23 percent of the observations are outliers. For the other data sets, the fraction of outliers was reduced to below 7 percent of all data.

The aspect of multicollinearity can be quantified using the Variance Inflation Factor (VIF), which is defined as

$$VIF_j = \frac{1}{1 - R_j^2},$$

with R_j^2 the coefficient of determination obtained by regressing $\mathbf{x}_j$ on all other independent attributes. A value higher than 5 is typically considered to be an indication of multicollinearity. Most data sets are characterized by limited multicollinearity; only Desharnais, USP05, ISBSG, and Euroclear data sets had a VIF higher than 5 for over

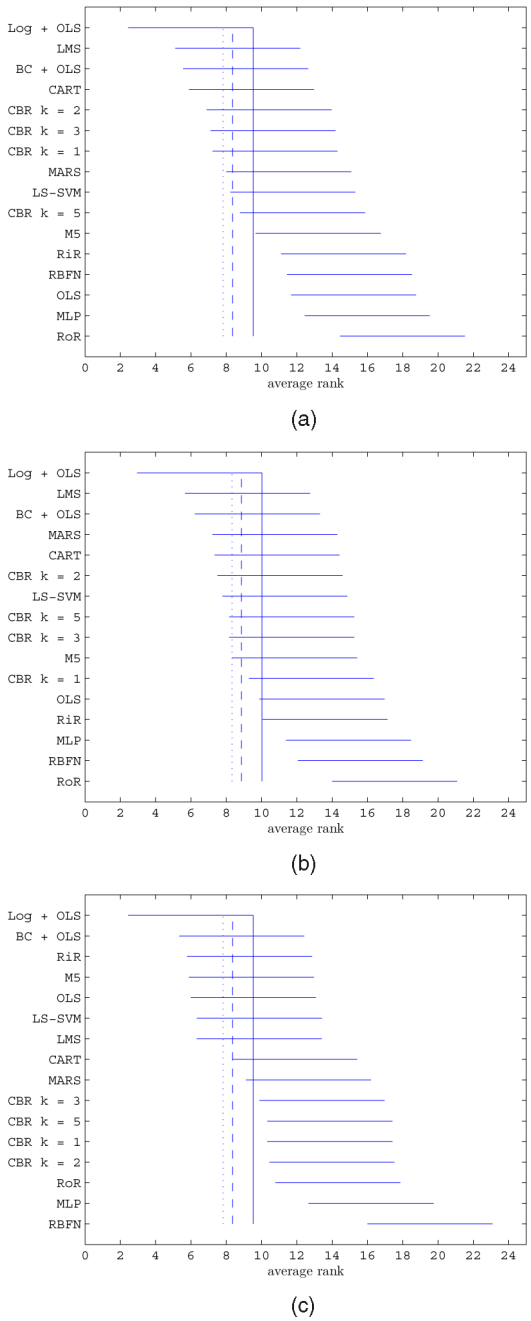


Fig. 3. Ranking of software effort estimation models without backward input selection for (a) MdMRE, (b) Pred_{25}, and (c) Spearman's rank correlation. The dotted vertical line represents the 90 percent significance level, the dashed line the 95 percent significance level, and the full line the 99 percent significance level. (a) Plot of the Bonferroni-Dunn test for MdMRE, (b) plot of the Bonferroni-Dunn test for Pred_{25}, (c) plot of the Bonferroni-Dunn test for Spearman's rank correlation.

50 percent of their attributes. Still, it can be remarked that in these cases, ridge regression, which is specifically designed to cope with multicollinearity, did not score particularly well. A possible explanation is that the regression models did not consider all attributes concurrently but instead used a stepwise selection procedure.

Finally, taking the number of observations into account, one would expect nonlinear techniques such as SVM, RBFN, and MLP to perform better in case of larger data sets. However, even in case of the largest data set (ISBSG), state-of-the-art nonlinear techniques did not perform particularly well. While LS-SVM did not perform statistically worse than Log + OLS, both MLP and RBFN were found to be outperformed. Both tree/rule induction techniques (CART and M5) were not statistically outperformed by Log + OLS. A possible explanation is that MLP, LS-SVM, and RBFN take all attributes jointly into account while other techniques consider only one attribute at the time. Hence, these other techniques are less affected by the sparseness of the data sets. Note that a total of 1,160 observations is not large compared to a number of other domains [17].

It should be noted that different performance metrics measure different properties of the distribution of $\hat{e}_i$ [86], and thus could give inconsistent results if they are used to evaluate alternative prediction models. Similar observations can be made in other studies that used multiple performance metrics, e.g., [21], [57].

The Pred_{25} metric favors techniques that are generally accurate (e.g., fall within 25 percent of the actual value) and occasionally widely inaccurate. The MdMRE is an analogous measure as it is also outlier resistant. Both can therefore be considered to be measures benefiting from properly calibrated models. This can be seen by the similar results for both the Pred_{25} and the MdMRE, see Figs. 3a and 3b. The results for the Spearman's rank correlation are slightly more deviant since, for instance, RiR scores third and model trees (M5) fourth, see Fig. 3c. The best and worst performing techniques are, however, similar. The Spearman's rank correlation is a measure of monotonic relationship between the actual and the predicted values and is therefore insensitive to the precise calibration of the models.

Focusing on the results in terms of MdMRE, cfr. Table 5, it can be seen that Log + OLS is the best performing technique as it is ranked first in seven of nine cases. Hence, the best average rank is attributed to Log + OLS, followed by LMS, BC + OLS, CART, various implementations of CBR, and MARS, none of which is significantly outperformed at the 95 percent significance level. The excellent results of various implementations of regression allow to build accurate and comprehensible software effort estimation models, which can be checked against prior domain knowledge. These findings are consistent with the studies of Briand et al. [14], [15], who found that OLS is a good performing technique on previous versions of both Experience and ESA data sets. From a business perspective, the aspects of understandability and trust are important, thus techniques resulting in comprehensible and justifiable models (i.e., are in line with generally accepted domain knowledge) are preferred [93]. The good result of CART, which is not outperformed by the best performing technique at the 95 percent significance level, is interesting since CART allows us to induce easy to understand piecewise linear prediction functions, and permits verifying whether the learned model is in line with prior domain knowledge, see Fig. 1d. Note, however, that models occasionally would need to be recalibrated on newly collected data, as relationships between attributes can change over time [19]. The fact that multiple techniques

TABLE 8
Results of the Cocomo Models

Data set	Technique	MdMRE	Pred_{25}	r_s
Cocnasa	Cocomo	**26.9**	**46.2**	**0.966**
	Log + OLS	30.4	45.2	0.889
Coc81	Cocomo	31.9	36.5	**0.991**
	Log + OLS	**27.6**	**49.2**	0.863

perform similar is in line with previous benchmarking studies in related domains like software fault prediction [94].

5.1.1 Cocomo

A comparison can be made between formal models such as Cocomo and data mining techniques. However, as indicated in Section 2, applying Cocomo requires a data set to be in a suitable format. Hence, only two data sets qualify for this comparison (the Coc81 and Cocnasa data sets). Similarly to data mining techniques which are typically tuned to better fit the underlying problem area, the coefficient a and the exponent b of the Cocomo model can be adjusted [7, chapter 29]. This requires the use of a separate training and test sample. Again, the same leave-one-out approach is followed to allow for better comparison to the other results, tuning both parameters on the complete data set minus one observation. The calibrated model is then used to estimate the effort of the test observation. Table 8 shows the result of this experiment. Comparing the Cocomo results to other data mining techniques shows that Cocomo yields similar results as regression with logarithmic transformation, which was previously found to be the best technique in both cases. Bold font indicates the best performing technique; no statistical significant differences were found by comparing the results between Cocomo and Log + OLS. Analogous results are to be expected as a Cocomo model is similar in nature to a regression model with logarithmic transformation.

It is interesting to note that the performance of Cocomo on the Coc81 data set can be partially attributed to the way the Cocomo model was constructed; the original Cocomo model was built and calibrated (including the precise values for each of the effort multipliers) on 56 of the 63 observations, with the (independent) test set consisting of 7 observations [7, chapter 29].

It should be noted that not all attributes are equally important in estimating software effort. Therefore, in the next section, the results of a backward input selection procedure are discussed.

5.2 Backward Input Selection

The lower panels of Tables 5, 6, and 7 show the results of the generic backward input selection procedure. A Friedman test to compare the results with backward input selection and without backward input selection is performed (in this case is k equals 2 and P equals 9), yielding a p-value close to zero (p-value < 0.02 in all three cases). This indicates that on aggregate applying input selection yields significantly higher performance. While this result might seem counterintuitive at first sight since information on certain attributes is removed from the data set, it makes sense that learning from a smaller data set, containing a limited set of highly predictive attributes, is easier than learning from a noisy data set containing many redundant and/or irrelevant attributes. The resulting models with input selection will also be more stable since potential collinearity between attributes will be reduced. Moreover, a model containing less attributes is generally preferred over a model with more attributes since a more concise model is easier to interpret. This finding is a confirmation of previous research, e.g., the work of Chen et al. [72], which found that a higher accuracy could be achieved using a wrapper approach in case of Cocomo, and the findings of Azzeh et al. [95], who found similar results in case of CBR on the Desharnais and the ISBSG data set. These findings were also confirmed by Li et al. [74], using a mutual information filter approach on the Desharnais and the Maxwell data set. Fig. 4 plots a typical result of the backward input selection procedure which is exemplary for most techniques and data sets. On the horizontal axis, the number of variables remaining in the model are given, while on the vertical axis, the performance measured in MdMRE is provided. The number of remaining attributes is selected by identifying the elbow point in the performance evolution of the different techniques. Fig. 5 provides box plots of the number of selected attributes. In each box plot, the central line indicates the median number of selected attributes, while the edges of the box represent the 25th and the 75th percentiles for each technique. The whiskers extend to the most extreme numbers of selected attributes that are not considered to be outliers. Outliers, finally, are represented by crosses.

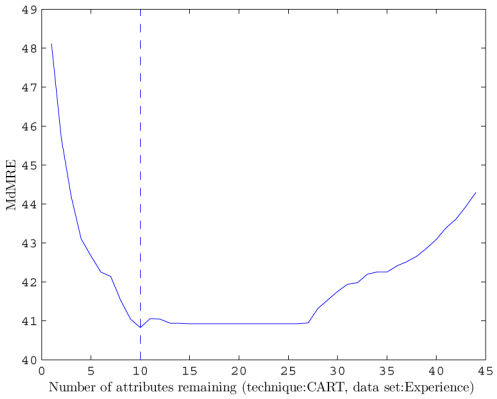


Fig. 4. Performance evolution of the input selection procedure for CART applied on the ISBSG data set.

5.2.1 Technique Evaluation

The same statistical procedure is followed as in Section 5.1. The results of the Friedman test indicate the existence of significant differences between techniques (p-values between 0.0000 and 0.0066). Subsequently, Bonferroni-Dunn tests are applied. The results of these tests are plotted in Fig. 6. From these plots, it can be concluded that in all three cases the best performing technique is again Log + OLS.

Analogous to the case without input selection, the results from both MdMRE and Pred_{25} are similar. The best performing technique is Log + OLS, followed closely by a number of other techniques including LMS, BC + OLS, MARS, LS-SVM, and various implementations of CBR. Again, more deviant results can be observed in case of

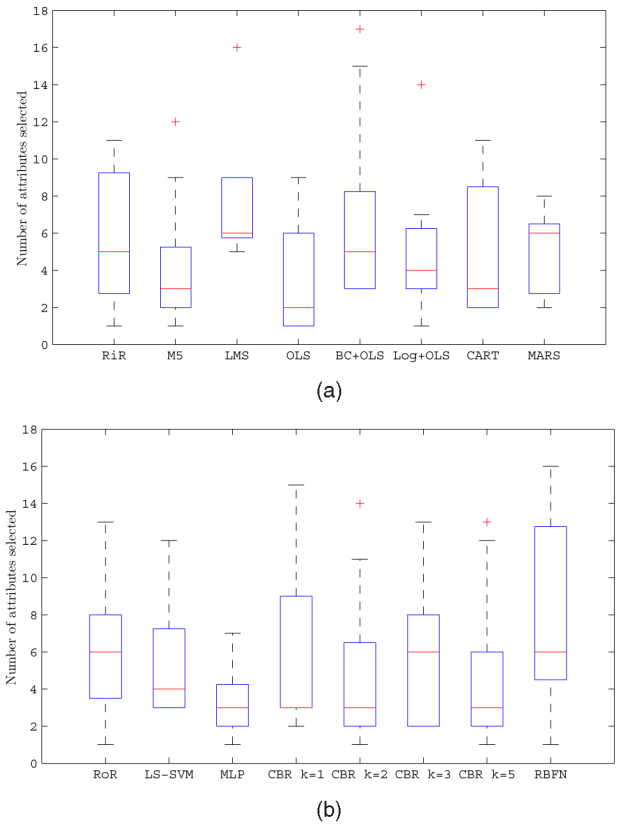


Fig. 5. Boxplot of the number of attributes selected by the different techniques. (a) Boxplots of RiR, M5, LMS, OLS, BC + OLS, Log + OLS, CART, and MARS, (b) Boxplots of RoR, LS-SVM, MLP, CBR $k = 1$, CBR $k = 2$, CBR $k = 3$, CBR $k = 5$, and RBFN.

Spearman's rank correlation, although the best performing technique is similar to both other metrics. Fig. 8 provides an example of an OLS + Log model after application of the generic backward input selection procedure on both the Experience and the ESA data set. An advantage of regression models is the possibility of verifying whether the model is in line with domain knowledge. For instance, it can be anticipated that larger projects require more effort, and thus a positive coefficient is expected. Similarly, in case of more programming experience, a lower effort is required and thus a negative coefficient is expected.

While the Friedman test indicates that, on average, the results are better with input selection, analogous conclusions can be drawn to which techniques are more or less suited for software effort estimation. Log + OLS is again overall the best performing technique, while a number of nonlinear techniques such as RBFN are less able to provide good estimations. Whether a specific technique on a specific data set benefits from selecting a subset of the data needs to be verified empirically.

5.2.2 Selected Attributes

It can be seen from Fig. 5 that the number of selected attributes by the different techniques is rather low, typically ranging from 2 to 10. This means that a surprisingly small number of attributes suffices to construct an effective software effort estimation model. Hence, the largest performance increase can be expected from improving the quality of data, instead of collecting more attributes of low-predictive value. Data quality is an important issue in the context of any data mining task [96], [97].

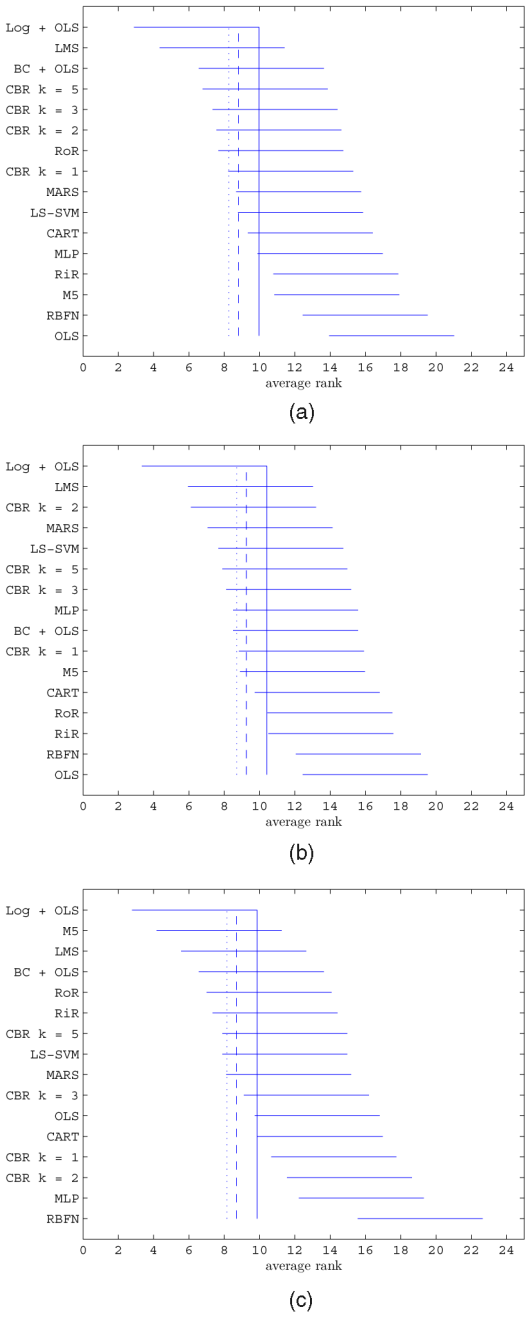


Fig. 6. Ranking of software effort estimation models with backward input selection for (a) MdMRE, (b) Pred_{25}, and (c) Spearman's rank correlation. The dotted vertical line represents the 90 percent significance level, the dashed line the 95 percent significance level, and the full line the 99 percent significance level. (a) Plot of the Bonferroni-Dunn test for MdMRE, (b) plot of the Bonferroni-Dunn test for Pred_{25}, (c) plot of the Bonferroni-Dunn test for Spearman's rank correlation.

In Section 4.1, four attribute types were identified that are present in the software effort estimation data sets: size, environment, project, and development. As a result of the input selection procedure, it is possible to identify the most important attribute types in the data sets, see Fig. 7. Size, development, and environment are considered to be

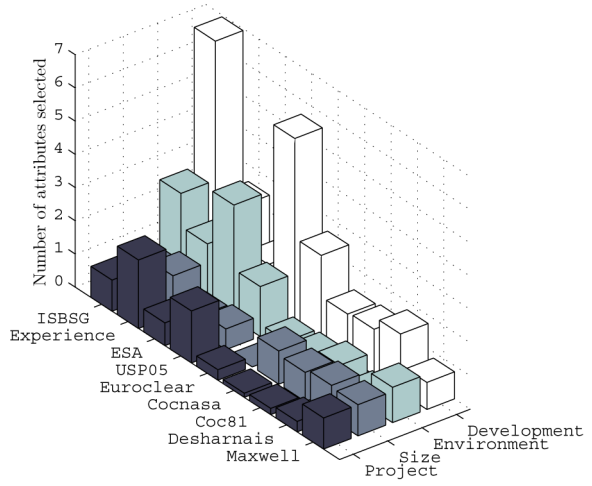


Fig. 7. Bar chart of the average number of selected attributes per data set and per attribute type.

attributes of high importance for software effort prediction. It should be noted that the attribute type "size" typically only covers a limited number of attributes in a data set. Since this type of attribute is selected in nearly all cases, it is therefore considered to be highly predictive.

Another observation that can be made, is that all four types of attributes are included in the set of most predictive attributes selected during the input selection procedure. Hence, none of the four attribute types can be omitted from the data sets without incurring some performance loss. Focusing on specific attribute categories, some attributes which are typically good predictors can be identified. For instance, programming language is a development-type attribute that is often selected by the input selection procedure. This is to be expected, since programming language was previously found to have an important impact on development effort, e.g., by Albrecht and Gaffney [10]. Concerning environment attributes, variables related to team size and company sector prove to be good predictors to estimate effort as well.

We also considered a minimal Redundancy, Maximum Relevance (mRMR) filter approach [98] as an alternative to the backward input selection in this study, similar to the study of Li et al. [74]. Using this approach, we selected the top 10 ranking attributes. This filter approach gave, however, similar results to the backward input selection approach, i.e., also indicating an increased performance by taking a highly predictive set of attributes and the importance of the size-related attribute. In order not to overload the paper, the results of this filter are not further detailed.

Data set / Obtained model
Experience - Reduced model
$\text{Log}(\hat{e}_i)$ = 3.71 + 0.72×log(size) + 0.12×(IT staff req.) - 0.09×(Usability req.) - 0.73×(Company2) - 0.42×(Company3)
ESA - Reduced model
$\text{Log}(\hat{e}_i)$ = 1.51 + 0.41×log(size) + 0.92×log(Team size) + 0.14×(Mem. constraint) + 0.55×(Fortran/AS) - 0.77×(UK) - 0.23×(Lang. experience) - 0.82×(ESA project category 3)

Fig. 8. Example of the best performing technique.

TABLE 9
Overview of Calibration and Computation Time

Technique	Over-estimate	Under-estimate	Aggregate	Comp. time
RiR	47.8	29.8	18.0	Medium
M5	44.1	32.9	11.2	Low
LMS	36.5	38.1	-1.6	Low
OLS	43.1	34.7	8.4	Low
BC + OLS	36.9	37.7	-0.8	Low
Log + OLS	33.5	31.4	2.1	Low
CART	47.5	27	20.5	Low
MARS	44.7	31.3	13.4	High
RoR	39.6	46	-6.4	Low
LS-SVM	45.8	28.6	17.1	High
MLP	51.5	33	18.5	High
CBR k = 1	37.3	38.3	-1	Low
CBR k = 2	41.4	33	8.4	Low
CBR k = 3	45.2	30	15.2	Low
CBR k = 5	46.2	28.6	17.6	Low
RBFN	33.6	49.1	-15.5	Low

6 Conclusions and Future Research

The results of this benchmarking study partially confirm the results of previous studies [14], [15], [21]. Simple, understandable techniques like OLS with log transformation of attributes and target perform as well as (or better than) nonlinear techniques. Additionally, a formal model such as Cocomo performed at least equally well as OLS with log transformation on the Coc81 and Cocnasa data sets. These two data sets were collected with the Cocomo model in mind. However, this model requires a specific set of attributes and cannot be applied on data sets that do not comply with this requirement. Although the performance differences can be small in absolute terms, a minor difference in estimation performance can cause more frequent and larger project cost overruns during software development. Hence, even small differences can be important from a cost and operational perspective [99].

These results also indicate that data mining techniques can make a valuable contribution to the set of software effort estimation techniques, but should not replace expert judgment. Instead, both approaches should be seen as complements. Depending on the situation, either expert-driven or analytical methods might be preferable as first line estimation. In case the experts possess a significant amount of contextual information not available to an analytical method, expert-driven approaches might be preferred [24]. An automated data mining technique can then be adopted to check for potential subjective biases in the expert estimations. Additionally, various estimations can be combined in alternative ways to improve the overall accuracy, as investigated by, e.g., MacDonell and Shepperd [100], which concluded "that there is indeed potential benefit in using more than one technique." A simple approach could be to take the average across estimations, while a more advanced approach would investigate in which case a specific technique yields the most accurate estimation. When combining estimates of techniques, the potential bias of the technique, i.e., the tendency to over or underestimate effort, should be taken into account. Since effort is a continuous attribute, typically some error is to be expected. However, if

the estimate is far from the actual value, e.g., more than 25 percent, the estimate can not be considered "accurate." The first two columns in Table 9 provide the average over and underestimation per technique across all data sets. The third column aggregates these two values, indicating whether a technique has an overall tendency to over or underestimate. Combining different techniques increases computational requirements and hampers comprehensibility. The required computation time is dependent on a number of different aspects, including data set, hardware, technique, empirical setup, and parameter tuning. An indication of computational requirements is also presented in the last column of Table 9.

A third conclusion is that the selection of a proper estimation technique can have a significant impact on the performance. A simple technique like regression is found to be well suited for software effort estimation which is particularly interesting as it is a well-documented technique with a number of interesting qualities like statistical significance testing of parameters and stepwise analysis. This conclusion is valid with respect to the different metrics that are used to evaluate the techniques. Furthermore, it is shown that, typically, a significant performance increase can be expected by constructing software effort estimation models with a limited set of highly predictive attributes. Hence, it is advised to focus on data quality rather than collecting as much predictive attributes as possible. Attributes related to the size of a software project, to the development, and to environment characteristics are considered to be the most important types of attributes.

6.1 Future Research

This study indicates that different data preprocessing steps, addressing possible data quality issues such as discretization algorithms, missing value handling schemas, and scaling of attributes, can play an important role in software effort estimation. While the same data preprocessing steps were applied on all data sets, the results of the input selection indicate that preprocessing steps such as attribute selection can be important. A thorough assessment of all possible data preprocessing steps seems, however, computationally infeasible when considering a large number of techniques and data sets. The impact of various preprocessing techniques has already been investigated to a limited extent (e.g., [66]), but further research into this aspect could provide important insights.

Considering the typical limited number of observations and the importance of expert knowledge (i.e., contextual information) for software effort estimation, we believe the inclusion of such expert knowledge to be a promising topic for future research. The inclusion of domain knowledge (e.g., by enforcing monotonicity constraints) into data mining models constitutes the so-called knowledge fusion problem [93]. Employing, e.g., hard constraints (constraints which should not be violated) to restrict the solution space can be an effective way to deal with the sparseness of data associated with the small number of observations [101]. The inclusion of domain knowledge is also important to both software engineers and management to inspire more confidence into such models, thus facilitating the introduction of these models into business settings.

Related hereto, is the aspect of comprehensibility of the estimation model as it also plays an important role in acceptance of the model in a business setting. While log-linear models are understandable to a certain level, rule sets or decision trees are considered more comprehensible to end users [102]. This topic has been investigated only to a limited extent in software effort estimation; see, e.g., [71]. Future research could be done into these aspects by, e.g., studying the framework of Andrews et al. [103], especially considering the impact such estimations can have on the budgeting and remuneration of staff.

APPENDIX

DETAILS DATA SELECTION

For each data set, similar data preprocessing rules were followed, as detailed in Section 4.1. The application of these rules is illustrated on the largest data set, the ISBSG R11 data set.

Details original data set	*Details preprocessed data set*
# Observations 5052	# Observations 1160
# Attributes 115	# Attributes 14

Attributes selection guidelines:

- Only attributes pertaining to software development are retained (i.e. attributes referring to software quality and software productivity are removed).
- Remove attributes counting towards a global attribute such as
  - value adjustment factor,
  - sizing attributes used in obtaining function point count.
- Remove attributes that are unknown at the moment of estimation such as project duration.
- Remove attributes with more than 25% missing values, such as
  - Software process CMMI: 97% missing,
  - 2^{nd} programming language: 96% missing,
  - Package customization: 44% missing.

Observation selection guidelines:

- Projects are retained with an overall data quality rating of A or B. B quality is defined as 'The submission appears to be fundamentally sound but there are some factors which could affect the integrity of the submitted data'.
- Retained projects with function point quality of A. B quality label indicates that 'the unadjusted function point count appears sound, but integrity cannot be assured since a single figure was provided'. This is potentially more problematic than a B rating for overall quality as size attributes are typically the most predictive attributes.
- Retained projects of which the function points are counted using the IFPUG 4 standard, as indicated in the ISBSG guidelines: 'You shouldn't mix pre-IFPUG V4 projects with V4 and post V4 (the sizing changed with that release)'.
- Retained projects of which effort refers to resource level 1 (i.e. only development team effort included).
- Retained projects with no missing value for categorical attribute 'primary programming language' as only 13% of the observations have a missing value for this attribute.

Missing value handling

- Median value imputation for variable 'Team size'. The imputed value is '6'.

Dummy encoding

- Dummy encoding nominal attributes
  - Development type
  - Organization type
  - Business area type
  - Application type
  - Architecture
  - Development technique
  - 1^{st} Database system
  - Platform
  - Language type
  - Primary programming language
  - 1^{st} Hardware
  - 1^{st} Operating system

ACKNOWLEDGMENTS

The authors would like to express their gratitude to Mr. Pekka Forselius, 4sumpartners, Mr. Benjamin Schreiber, ESA, and Mr. Laurent Fayet, Euroclear, for providing valuable data that made this research possible. They are also indebted to Jochen De Weerdt and Helen Moges for providing computational support during the analysis. They extend their gratitude to the Flemish Research Fund for financial support to the authors (Odysseus grant G.0915.09 to Bart Baesens and postdoctoral research grant to David Martens).

REFERENCES

[1] The Standish Group, "Chaos Report," Technical report, http://www.standishgroup.com, 2009.
[2] M. Jørgensen and M. Shepperd, "A Systematic Review of Software Development Cost Estimation Studies," *IEEE Trans. Software Eng.,* vol. 33, no. 1, pp. 33-53, Jan. 2007.
[3] M. Jørgensen and S. Wallace, "Improving Project Cost Estimation by Taking into Account Managerial Flexibility," *European J. Operational Research,* vol. 127, pp. 239-251, 2000.
[4] E.A. Nelson, *Management Handbook for the Estimation of Computer Programming Costs.* System Developer Corp., 1966.
[5] B. Kitchenham, S. Pfleeger, B. McColl, and S. Eagan, "An Empirical Study of Maintenance and Development Estimation Accuracy," *The J. Systems and Software,* vol. 64, pp. 57-77, 2002.
[6] M. Jørgensen, "A Review of Studies on Expert Estimation of Software Development Effort," *The J. Systems and Software,* vol. 70, pp. 37-60, 2004.
[7] B. Boehm, *Software Engineering Economics.* Prentice Hall, 1981.
[8] B. Boehm, R. Madachy, and B. Steece, *Software Cost Estimation with Cocomo II.* Prentice Hall, 2000.
[9] L.H. Putnam, "A General Empirical Solution to the Macro Software Sizing and Estimation Problem," *IEEE Trans. Software Eng.,* vol. 4, no. 4, pp. 345-361, July 1978.
[10] A.J. Albrecht and J.E. Gaffney, "Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation," *IEEE Trans. Software Eng.,* vol. 9, no. 6, pp. 639-648, Nov. 1983.
[11] C.F. Kemerer, "An Empirical Validation of Software Cost Estimation Models," *Comm. ACM,* vol. 30, no. 5, pp. 416-429, 1987.
[12] G. Finnie, G. Wittig, and J.-M. Desharnais, "A Comparison of Software Effort Estimation Techniques: Using Function Points with Neural Networks, Case-Based Reasoning and Regression Models," *J. Systems and Software,* vol. 39, pp. 281-289, 1997.
[13] P. Sentas, L. Angelis, I. Stamelos, and G. Bleris, "Software Productivity and Effort Prediction with Ordinal Regression," *Information and Software Technology,* vol. 47, pp. 17-29, 2005.
[14] L. Briand, K.E. Emam, D. Surmann, and I. Wieczorek, "An Assessment and Comparison of Common Software Cost Estimation Modeling Techniques," *Proc. 21st Int'l Conf. Software Eng.,* pp. 313-323, May 1999.
[15] L. Briand, T. Langley, and I. Wieczorek, "A Replicated Assessment and Comparison of Common Software Cost Modeling Techniques," *Proc. 22nd Int'l Conf. Software Eng.,* pp. 377-386, June 2000.
[16] J. Li, G. Ruhe, A. Ak-Emran, and M. Richter, "A Flexible Method for Software Effort Estimation by Analogy," *Empirical Software Eng.,* vol. 12, pp. 65-107, 2007.
[17] B. Baesens, T.V. Gestel, S. Viaene, M. Stepanova, J. Suykens, and J. Vanthienen, "Benchmarking State-of-the-Art Classification Algorithms for Credit Scoring," *J. Operational Research Soc.,* vol. 54, no. 6, pp. 627-635, 2003.
[18] W. Verbeke, D. Martens, C. Mues, and B. Baesens, "Building Comprehensible Customer Churn Prediction Models with Advanced Rule Induction Techniques," *Expert Systems with Applications,* vol. 38, pp. 2354-2364, 2011.
[19] B. Kitchenham and E. Mendes, "Why Comparative Effort Prediction Studies May Be Invalid," *Proc. Fifth Int'l Conf. Predictor Models in Software Eng.,* 2009.
[20] M. Shepperd and C. Schofield, "Estimating Software Project Effort Using Analogies," *IEEE Trans. Software Eng.,* vol. 23, no. 12, pp. 736-743, Nov. 1997.
[21] I. Myrtveit and E. Stensrud, "A Controlled Experiment to Assess the Benefits of Estimation with Analogy and Regression Models," *IEEE Trans. Software Eng.,* vol. 25, no. 4, pp. 510-525, July/Aug. 1999.
[22] B. Littlewood, P. Popov, and L. Strigini, "Modeling Software Design Diversity a Review," *ACM Computing Surveys,* vol. 33, no. 2, pp. 177-208, 2001.
[23] R.M. Dawes, D. Faust, and P.E. Meehl, "Clinical versus Actuarial Judgement," *Science,* vol. 243, no. 4899, pp. 1668-1674, 1989.
[24] M. Jørgensen, "Forecasting of Software Development Work Effort: Evidence on Expert Judgement and Formal Models," *Int'l J. Forecasting,* vol. 23, pp. 449-462, 2007.
[25] T. Mukhopadhyay, S.S. Vicinanza, and M.J. Prietula, "Examining the Feasibility of a Case-Based Reasoning Model for Software Effort Estimation," *MIS Quarterly,* vol. 16, no. 2, pp. 155-171, 1992.
[26] M. Jørgensen and D.I.K. Søberg, "Impact of Experience on Maintenance Skills," *J. Software Maintenance and Evolution: Research and Practice,* vol. 14, no. 2, pp. 123-146, 2002.
[27] F.J. Heemstra and R.J. Kusters, "Function Point Analysis: Evaluation of a Software Cost Estimation Model," *European J. Information Systems,* vol. 1, no. 4, pp. 223-237, 1991.
[28] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning, Data Mining, Inference, and Prediction.* Springer, 2001.
[29] G. Box and D. Cox, "An Analysis of Transformations," *J. Royal Statistical Soc., Series B,* vol. 26, pp. 211-252, 1964.
[30] R. Sakia, "The Box-Cox Transformation Technique: A Review," *J. Royal Statistical Soc. Series D,* vol. 41, no. 2, pp. 169-178, 1992.
[31] P.W. Holland, "Robust Regression Using Iteratively Reweighted Least Squares," *Comm. in Statistics: Theory and Methods,* vol. 6, no. 9, pp. 813-827, 1977.
[32] C. Moler, "Society for Industrial and Applied Mathematics," *Numerical Computing with Matlab,* Chapter 5, pp. 1-27, Cambridge Univ. Press, 2004.
[33] R. Jeffery, M. Ruhe, and I. Wieczorek, "Using Public Domain Metrics to Estimate Software Development Effort," *Proc. Seventh Int'l Software Metrics Symp.,* pp. 16-27, Apr. 2001.
[34] A.E. Hoerl, "Application of Ridge Analysis to Regression Problems," *Chemical Eng. Progress,* vol. 58, no. 5, pp. 54-59, 1962.
[35] P. Rousseeuw, "Least Median of Squares Regression," *J. Am. Statistical Assoc.,* vol. 79, no. 388, pp. 871-880, 1984.
[36] F.R. Hampel, "A General Qualitative Definition of Robustness," *Annals of Math. Statistics,* vol. 42, pp. 1887-1896, 1971.
[37] J.H. Friedman, "Multivariate Adaptive Regression Splines," *Annals of Statistics,* vol. 19, no. 1, pp. 1-67, 1991.
[38] T.-S. Lee and I.-F. Chen, "A Two-Stage Hybrid Credit Scoring Model Using Artificial Neural Networks and Multivariate Adaptive Regression Splines," *Expert Systems with Applications,* vol. 28, pp. 743-752, 2005.
[39] J. Elith and J. Leathwick, "Predicting Species Distributions from Museum and Herbarium Records Using Multiresponse Models Fitted with Multivariate Adaptive Regression Splines," *Diversity and Distributions,* vol. 13, no. 3, pp. 265-275, 2007.
[40] J.R. Quinlan, *C4.5: Programs for Machine Learning.* Morgan Kaufmann, 2003.
[41] L. Breiman, J.H. Friedman, R.A. Olsen, and C.J. Stone, *Classification and Regression Trees.* Wadsworth & Books/Cole Advanced Books & Software, 1984.
[42] B. Kitchenham, "A Procedure for Analyzing Unbalanced Data Sets," *IEEE Trans. Software Eng.,* vol. 24, no. 4, pp. 278-301, Apr. 1998.
[43] E. Mendes, I. Watson, C. Triggs, N. Mosley, and S. Counsell, "A Comparative Study of Cost Estimation Models for Web Hypermedia Applications," *Empirical Software Eng.,* vol. 8, pp. 163-196, 2003.
[44] J.R. Quinlan, "Learning with Continuous Classes," *Proc. Fifth Australian Joint Conf. Artificial Intelligence,* Adams and Sterling, eds., pp. 343-348, 1992.
[45] Y. Wang and I.H. Wittig, "Induction of Model Trees for Predicting Continuous Classes," *Proc. Poster Papers Ninth European Conf. Machine Learning,* 1997.
[46] C.M. Bishop, *Neural Networks for Pattern Recognition.* Oxford Univ. Press, 1995.
[47] J.M. Zurada, *Introduction to Artificial Neural Systems.* PWS Publishing Company, 1995.
[48] B.D. Ripley, *Pattern Recognition and Neural Networks.* Cambridge Univ. Press, 1996.

[49] C. Burgess and M. Lefley, "Can Genetic Programming Improve Software Effort Estimation? A Comparative Evaluation," *Information and Software Technology,* vol. 43, pp. 863-873, 2001.

[50] M. Lefley and M. Shepperd, "Using Genetic Programming to Improve Software Effort Estimation Based on General Data Sets," *Proc. Int'l Conf. Genetic and Evolutionary Computation,* pp. 2477-2487, 2003.

[51] K. Hornik, M. Stinchcombe, and H. White, "Multilayer Feedforward Networks Are Universal Approximators," *Neural Networks,* vol. 2, no. 5, pp. 359-366, 1989.

[52] M.T. Hagan and M.B. Menhaj, "Training Feedforward Networks with the Marquardt Algorithm," *IEEE Trans. Neural Networks,* vol. 5, no. 6, pp. 989-993, Nov. 1994.

[53] J. Moody and C. Darken, "Fast Learning in Networks of Locally-Tuned Processing Units," *Neural Computing,* vol. 1, pp. 281-294, 1989.

[54] D. Specht, "A General Regression Neural Network," *IEEE Trans. Neural Networks,* vol. 2, no. 6, pp. 568-576, Nov. 1991.

[55] A. Idri, A. Zahi, E. Mendes, and A. Zakrani, "Software Cost Estimation Models Using Radial Basis Function Neural Networks," *Software Process and Product Measurement,* pp. 21-31, Springer-Verlag, 2008.

[56] A. Heiat, "Comparison of Artificial Neural Networks and Regression Models for Estimating Software Development Effort," *Information and Software Technology,* vol. 44, no. 15, pp. 911-922, 2002.

[57] N.-H. Chiu and S.-J. Huang, "The Adjusted Analogy-Based Software Effort Estimation Based on Similarity Distances," *The J. Systems and Software,* vol. 80, pp. 628-640, 2007.

[58] V.N. Vapnik, *Statistical Learning Theory.* John Wiley, 1998.

[59] J. Suykens and J. Vandewalle, "Least Squares Support Vecrot Machine Classifiers," *Neural Processing Letters,* vol. 9, no. 3, pp. 293-300, 1999.

[60] T. Van Gestel, J. Suykens, B. Baesens, S. Viaene, J. Vanthienen, G. Dedene, B.D. Moor, and J. Vandewalle, "Benchmarking Least Squares Support Vector Machine Classifiers," *Machine Learning,* vol. 54, pp. 5-32, 2004.

[61] P. Rao, *Nonparametric Functional Estimation.* Academic Press, 1983.

[62] V. Kumar, V. Ravi, M. Carr, and R. Kiran, "Software Development Cost Estimation Using Wavelet Neural Networks," *The J. Systems and Software,* vol. 81, pp. 1853-1867, 2008.

[63] C. Mair, M. Shepperd, and M. Jorgensen, "An Analysis of Datasets Used to Train and Validate Cost Prediction Systems," *ACM SIGSOFT Software Eng. Notes,* vol. 4, pp. 1-6, 2005.

[64] M. Auer, A. Trendowicz, B. Graser, E. Haunschmid, and S. Biffl, "Optimal Project Feature Selection Weigths in Analogy-Based Cost Estimation: Improvement and Limitations," *IEEE Trans. Software Eng.,* vol. 32, no. 2, pp. 83-92, Feb. 2006.

[65] K. Maxwell, L. Van Wassenhove, and S. Dutta, "Software Development Productivity of European Space, Military, and Industrial Applications," *IEEE Trans. Software Eng.,* vol. 22, no. 10, pp. 706-718, Oct. 1996.

[66] K. Strike, K.E. Emam, and N. Madhavji, "Software Cost Estimation with Incomplete Data," *IEEE Trans. Software Eng.,* vol. 27, no. 10, pp. 890-908, Oct. 2001.

[67] K. Maxwell, L. Van Wassenhove, and S. Dutta, "Performance Evaluation of General and Company Specific Models in Software Development Effort Estimation," *Management Science,* vol. 45, pp. 787-803, 1999.

[68] R. Jeffery, M. Ruhe, and I. Wieczorek, "A Comparative Study of Two Software Development Cost Modeling Techniques Using Multi-Organizational and Company-Specific Data," *Information and Software Technology,* vol. 42, no. 14, pp. 1009-1016, 2000.

[69] J. Li and G. Ruhe, "A Comparative Study of Attribute Weighting Heuristics for Effort Estimation by Analogy," *Proc. ACM-IEEE Int'l Symp. Empirical Software Eng.,* Sept. 2006.

[70] T. Menzies, Z. Chen, J. Hihn, and K. Lum, "Selecting Best Practices for Effort Estimation," *IEEE Trans. Software Eng.,* vol. 32, no. 11, pp. 883-895, Nov. 2006.

[71] A. Idri, T.M. Khoshgoftaar, and A. Abran, "Can Neural Networks Be Easily Interpreted in Software Cost Estimation?" *Proc. IEEE Int'l Conf. Fuzzy Systems,* pp. 1162-1167, 2002.

[72] Z. Chen, T. Menzies, D. Port, and B. Boehm, "Feature Subset Selection Can Improve Software Cost Estimation Accuracy," *ACM SIGSOFT Software Eng. Notes,* vol. 30, no. 4, pp. 1-6, 2005.

[73] M. Shepperd, C. Schofield, and B. Kitchenham, "Effort Estimation using Analogies," *Proc. 18th Int'l Conf. Software Eng.,* 1996.

[74] Y. Li, M. Xie, and T. Goh, "A Study of Mutual Information Based Feature Selection for Case Based Reasoning in Software Cost Estimation," *Expert Systems with Applications,* vol. 36, pp. 5921-5931, 2009.

[75] A. Tosun, B. Turhan, and A.B. Bener, "Feature Weighting Heuristics for Analogy-Based Effort Estimation Models," *Expert Systems with Applications,* vol. 36, pp. 10 325-10 333, 2009.

[76] J.M. Desharnais, "Analyse Statistique de la Productivities des Projets de Developpement en Informatique Apartir de la Techniques des Points de Fonction," PhD dissertation, Univ. du Quebec, 1988.

[77] K. Maxwell, *Applied Statistics for Software Managers.* Prentice-Hall, 2000.

[78] T. Van Gestel, B. Baesens, P. Van Dijcke, J. Garcia, J. Suykens, and J. Vanthienen, "A Process Model to Develop an Internal Rating System: Sovereign Credit Ratings," *Decision Support Systems,* vol. 42, no. 2, pp. 1131-1151, 2006.

[79] C. Kirsopp and M. Shepperd, "Making Inferences with Small Numbers of Training Sets," *IEE Proc. Software,* vol. 149, no. 5, pp. 123-130, Oct. 2002.

[80] R. Kohavi, "A Study on Cross-validation and Bootstrap for Accuracy Estimation and Model Selection," *Proc. 14th Int'l Joint Conf. Artificial Intelligence,* pp. 1137-1145, 1995.

[81] I. Myrtveit, E. Stensrud, and M. Shepperd, "Reliability and Validity in Comparative Studies of Software Prediction Models," *IEEE Trans. Software Eng.,* vol. 31, no. 5, pp. 380-391, May 2005.

[82] D. Martens, M. De Backer, R. Haesen, M. Snoeck, J. Vanthienen, and B. Baesens, "Classification with Ant Colony Optimization," *IEEE Trans. Evolutionary Computing,* vol. 11, no. 5, pp. 651-665, Oct. 2007.

[83] S.D. Conte, H.E. Dunsmore, and V.Y. Shen, *Software Engineering Metrics and Models.* The Benjamin/Cummings Publishing Company, Inc., 1986.

[84] D. Port and M. Korte, "Comparative Studies of the Model Evaluation Criterions MMRE and PRED in Software Cost Estimation Research," *Proc. Second ACM-IEEE Int'l Symp. Empirical Software Eng. and Measurement,* pp. 51-60, Oct. 2008.

[85] A. Miyazaki, A. Takanou, H. Nozaki, N. Nakagawa, and K. Okada, "Method to Estimate Parameter Values in Software Prediction Models," *Information and Software Technology,* vol. 33, no. 3, pp. 239-243, 1991.

[86] B. Kitchenham, L. Pickard, S. MacDonell, and M. Shepperd, "What Accuracy Statistics Really Measure," *IEE Proc. Software,* vol. 148, no. 3, pp. 81-85, June 2001.

[87] T. Foss, E. Stensrud, B. Kitchenham, and I. Myrtveit, "A Simulation Study of the Model Evaluation Criterion MMRE," *IEEE Trans. Software Eng.,* vol. 29, no. 11, pp. 985-995, Nov. 2003.

[88] J. Demsar, "Statistical Comparison of Classifiers over Multiple Data Sets," *J. Machine Learning Research,* vol. 7, pp. 1-30, 2006.

[89] M. Friedman, "A Comparison of Alternative Tests of Significance for the Problem of m Rankings," *Annals of Math. Statistics,* vol. 11, pp. 86-92, 1940.

[90] O.J. Dunn, "Multiple Comparisons among Means," *J. Am. Statistical Assoc.,* vol. 56, pp. 52-64, 1961.

[91] F. Wilcoxon, "Individual Comparisons by Ranking Methods," *Biometrics,* vol. 1, pp. 80-83, 1945.

[92] M. Shepperd and G. Kadoda, "Using Simulation to Evaluate Prediction Techniques," *Proc. Seventh Int'l Software Metrics Symp.,* pp. 349-359, 2002.

[93] D. Martens, M. De Backer, R. Haesen, B. Baesens, C. Mues, and J. Vanthienen, "Ant Based Approach to the Knowledge Fusion Problem," *Proc. Fifth Int'l Workshop Ant Colony Optimisation and Swarm Intelligence,* M. Dorigo, L. Gambardella, M. Birattari, A. Martinoli, R. Poli, and T. Stützle, eds., pp. 84-95, 2006.

[94] S. Lessmann, B. Baesens, C. Mues, and S. Pietsch, "Benchmarking Classification Models for Software Defect Prediction: A Proposed Framework and Novel Findings," *IEEE Trans. Software Eng.,* vol. 34, no. 4, pp. 485-496, July/Aug. 2008.

[95] M. Azzeh, D. Neagu, and P. Cowling, "Improving Analogy Software Effort Estimation Using Fuzzy Feature Subset Selection Algorithm," *Proc. Fourth Int'l Workshop Predictor Models in Software Eng.,* pp. 71-78, 2008.

[96] P.-N. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining.* Addison Wesley, 2005.

[97] B. Baesens, C. Mues, D. Martens, and J. Vanthienen, "50 Years of Data Mining and OR: Upcoming Trends and Challenges," *J. the Operational Research Soc.*, vol. 60, pp. 16-23, 2009.
[98] H. Peng, F. Long, and C. Ding, "Feature Selection Based on Mutual Information: Criteria of Max-Dependency, Max-Relevance, and Min-Redundancy," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 27, no. 8, pp. 1226-1238, Aug. 2005.
[99] M. Jørgensen and K. Moløkken-Østvold, "How Large Are Software Cost Overruns? A Review of the 1994 CHAOS Report," *Information and Software Technology*, vol. 48, pp. 297-301, 2006.
[100] S. MacDonell and M. Shepperd, "Combining Techniques to Optimize Effort Predictions in Software Project Management," *The J. Systems and Software*, vol. 66, pp. 91-98, 2003.
[101] E. Altendorf, E. Restificar, and T. Dietterich, "Learning from Sparse Data by Exploiting Monotonicity Constraints," *Proc. 21st Conf. Uncertainty in Artificial Intelligence*, pp. 18-26, 2005.
[102] J. Huysmans, K. Dejaeger, C. Mues, J. Vanthienen, and B. Baesens, "An Empirical Evaluation of the Comprehensibility of Decision Table, Tree and Rule Based Predictive Models," *Decision Support Systems*, vol. 51, pp. 141-154, 2011.
[103] R. Andrews, J. Diederich, and A. Tickle, "Survey and Critique of Techniques for Extracting Rules from Trained Artificial Neural Networks," *Knowledge-Based Systems*, vol. 8, no. 6, pp. 373-389, 1995.
[104] K. Srinivasan and D. Fisher, "Machine Learning Approaches to Estimating Software Development Effort," *IEEE Trans. Software Eng.*, vol. 21, no. 2, pp. 126-137, Feb. 1995.
[105] G. Wittig and G. Finnie, "Estimating Software Development Effort with Connectionist Models," *Information and Software Technology*, vol. 39, no. 7, pp. 469-476, 1997.
[106] S.-J. Huang, N.-H. Chiu, and L.-W. Chen, "Integration of the Grey Relational Analysis with Genetic Algorithm for Software Estimation," *European J. Operational Research*, vol. 188, pp. 898-909, 2008.
[107] H. Park and S. Baek, "An Empirical Validation of a Neural Network Model for Software Effort Estimation," *Expert Systems with Applications*, vol. 35, pp. 929-937, 2008.
[108] S. Koch and J. Mitlöhner, "Software Project Effort Estimation with Voting Rules," *Decision Support Systems*, vol. 46, pp. 895-901, 2009.

Karel Dejaeger graduated in 2009 as a business engineer from the Department of Decision Sciences and Information Management at the Katholieke Universiteit Leuven (K.U. Leuven). He is currently employed as a doctoral researcher at K.U. Leuven. His research interests include data mining, fraud detection, and software engineering.

Wouter Verbeke graduated in 2007 as a civil engineer from the Katholieke Universiteit Leuven (K.U. Leuven). He is currently working as a doctoral researcher on the Faculty of Business and Economics at the K.U. Leuven, Department of Decision Sciences and Information Management. His doctoral research focuses on the development and application of data mining and network modeling techniques in business settings, such as customer churn prediction models in the telco sector, and credit rating migration models in the financial sector.

David Martens received the PhD degree in applied economic sciences from the Department of Decision Sciences and Information Management of the Katholieke Universiteit Leuven (K.U. Leuven), Belgium, in 2008 and also received the master's degree in civil engineering from the Computer Science Department at K.U. Leuven in 2003, and a Master of Business Administration in 2005 from Reims Management School, France. He is an assistant professor at the University of Antwerp. His research is mainly focused on the development of comprehensible data mining techniques.

Bart Baesens is an associate professor at the Katholieke Universiteit Leuven (K.U. Leuven), Belgium, and a lecturer at the University of Southampton, United Kingdom. He has done extensive research on predictive analytics, data mining, customer relationship management, fraud detection, and credit risk management. His findings have been published in well-known international journals and presented at international top conferences. He is also a coauthor of the book *Credit Risk Management: Basic Concepts*, published in 2008.