Information and Software Technology 94 (2018) 1–13

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

Duplex output software effort estimation model with self-guided interpretation

Solomon Mensah[a,*], Jacky Keung[a], Michael Franklin Bosu[b], Kwabena Ebo Bennin[a]

[a] Department of Computer Science, City University of Hong Kong, Hong Kong, China
[b] Centre for Business, Information Technology and Enterprise, Wintec, Hamilton, New Zealand

ARTICLE INFO

Keywords:
Duplex output
Effort estimation
Effort classification
Multiple regression models

ABSTRACT

Context: Software effort estimation (SEE) plays a key role in predicting the effort needed to complete software development task. However, the *conclusion instability* across learners has affected the implementation of SEE models. This instability can be attributed to the lack of an *effort classification* benchmark that software researchers and practitioners can use to facilitate and interpret prediction results.
Objective: To ameliorate the *conclusion instability* challenge by introducing a classification and self-guided interpretation scheme for SEE.
Method: We first used the density quantile function to discretise the effort recorded in 14 datasets into three classes (*high, low* and *moderate*) and built regression models for these datasets. The results of the regression models were an effort estimate, termed *output 1*, which was then classified into an effort class, termed *output 2*. We refer to the models generated in this study as *duplex output models* as they return two outputs. The introduced *duplex output models* trained with the leave-one-out cross validation and evaluated with MAE, BMMRE and adjusted R^2, can be used to predict both the software effort and the class of software effort estimate. Robust statistical tests (Welch's *t*-test and Kruskal-Wallis *H*-test) were used to examine the statistical significant differences in the models' prediction performances.
Results: We observed the following: (1) the *duplex output models* not only predicted the effort estimates, they also offered a guide to interpreting the effort expended; (2) incorporating the genetic search algorithm into the *duplex output model* allowed the sampling of relevant features for improved prediction accuracy; and (3) ElasticNet, a hybrid regression, provided superior prediction accuracy over the ATLM, the state-of-the-art baseline regression.
Conclusion: The results show that the *duplex output model* provides a self-guided benchmark for interpreting estimated software effort. ElasticNet can also serve as a baseline model for SEE.

1. Introduction

Software effort estimation (SEE) is the process of predicting the effort needed for scheduling, costing and allocating resources to meet project delivery deadlines [1]. According to Menzies et al. [2], accurately predicting the software effort of a *new* project plays a significant role in software engineering, as it minimises overestimation and underestimation [3,4]. Both overestimating and underestimating software effort can result in project cancellation due to budget overruns [2] and can delay funding of project development [5]. In contrast, successful and accurate SEE leads to effective scheduling, monitoring and management of projects [6,7].

Although many prediction models have been developed [5,6,8,9–16,1] to aid in estimating software effort, challenges still exist in relation to the interpretation of the level of effort expended during development [17,7,18]. To the best of our knowledge, there is no benchmark to interpret and validate levels of estimated effort expended in SEE [17,19]. Irrespective of the complexity or simplicity of existing SEE models, the effort estimates mostly contradict the true or actual software effort shown in the validation sets [20]. This phenomenon is the result of the variations within SEE data, the selection criteria for data collection and the training and testing criteria used to set up the SEE models. Currently, there is no optimal SEE model due in part to data uncertainty and to variations within the training and validation sets [9]. The variation in estimation results from different SEE models leads to *conclusion instability* [7,18,21,19]. Thus, a SEE model without a corresponding interpretation guide might not be useful to software engineers. Therefore, there is a need for a more efficient and reliable

* Corresponding author.
E-mail addresses: smensah2-c@my.cityu.edu.hk (S. Mensah), Jacky.Keung@cityu.edu.hk (J. Keung), michael.bosu@wintec.ac.nz (M.F. Bosu), kebennin2-c@my.cityu.edu.hk (K.E. Bennin).

http://dx.doi.org/10.1016/j.infsof.2017.09.010
Received 5 December 2016; Received in revised form 19 September 2017; Accepted 22 September 2017
Available online 23 September 2017
0950-5849/

prediction model that not only estimates the software effort, but also offers a means through which the effort estimates can be effectively applied to specific contexts.

According to Keung et al. [7], the elusiveness of accurate prediction models for software effort has been a challenge for researchers, software engineers, practitioners, project managers and developers, who need an appropriate model and reliable training and validation sets to make relatively effective predictions of SEE [18]. Despite the numerous prediction models [9,6,22], the SEE research community still faces difficulties in making reliable and accurate predictions due to the following factors: (1) variations within SEE datasets, (2) the methodological procedures used for generating the training and validation sets, (3) the type of SEE models used and (4) the performance evaluation measures used.

Previous studies [2,6,7,18,23,12,10] have shown that these factors affect the prediction accuracy of SEE models. As poor SEE negatively affects the prediction accuracy and interpretation of results [24,7], there is a need to improve the *conclusion instability* of the prediction estimates. Previous studies [7,18,9,23,12,24,25] have shown that the *conclusion instability* of prediction estimates is significantly affected by the heterogeneous nature of the data, variations within SEE datasets and the methodological approach used to sample training sets. *Conclusion instability* refers to the variations in estimated results for a given dataset from different prediction models [7,18]. In this study, we focus on improving the *conclusion instability* of prediction estimates by introducing a software effort classification scheme to aid in the interpretation of estimation results in the context of the particular dataset and modelling approach used.

This study is motivated by the *conclusion instability* issue raised by Keung et al. [7], whereby inconsistent estimation results can be enhanced based on the selection of relevant training and validation sets, data preprocessing, the use of multiple prediction models and multiple evaluation measures. The conclusion instability of estimated results is an urgent problem in SEE, as there is no accepted prediction model to provide a precise and accurate prediction irrespective of the training and validation sets considered. We argue that classifying the continuous dependent variable (i.e. software effort) into discretised classes minimises the *conclusion instability*. Our classification of software effort can provide a benchmark to help identify the level of effort (*high, low* and *moderate*) expended during a software project development. We define *software effort classification* as the process of discretising the software effort of the training set into classes that will allow self-guided interpretation of the estimated results. Thus, for all training sets, we first discretise the continuous dependent variable (software effort) into three classes based on the density quantile function [26]. The classification of software effort is an auxiliary discretised dependent variable (*software effort class*) in addition to the main continuous dependent variable (*software effort*). These two dependent variables, referred to as the *duplex variables,* are used in addition to the independent variables in each dataset to set up the estimation models. We describe the estimation of the *duplex* output variables as *duplex output software effort estimation.*

Thus, in this study, we do not only estimate the continuous software effort as done in all existing SEE studies, but go a step further to classify the estimated effort according to the level of effort expended during software development. We hypothesise that these classification levels – *high, moderate* and *low* – can address the *conclusion instability* issue inherent in prediction modelling using SEE models. As different prediction models give different estimates, it is important for software engineers to consider the classification of *software effort* as an auxiliary prediction variable that can help them to select the best prediction model. According to Menzies et al. [2], some prediction models are most suitable for particular datasets. We argue that the classification scheme for *software effort* introduced in this study can address this issue. This is done by applying six *duplex output* SEE models to 14 SEE datasets and evaluating their prediction performances based on the estimated effort classifications.

It is problematic to use only prediction accuracy to determine the best and most stable SEE model, because the accuracy value is a single number and thresholds for better performance vary with the type of models, the datasets and the experimental settings. Classification schemes have already been used in the prediction of frauds and defects. We believe that the use of a classification scheme with prediction accuracy will aid in establishing a robust method for addressing the *conclusion instability* problem of SEE models. The classification scheme makes it easier to interpret the prediction accuracy and provides scope for the use of the software effort estimates.

Software engineers are interested in the accuracy of estimation results from prediction models. We therefore argue that to obtain a more accurate and trustworthy estimate, it is necessary to predict the *effort classes*. When *software effort* is classified, we can easily compare and contrast the prediction accuracy of multiple SEE models, irrespective of the evaluation measure used for a given dataset.

According to Mair et al. [27], there exists difficulty in obtaining relevant datasets for studying effort estimation, as the appropriate ones might be proprietary. Biasness in the project dataset selection and model training criteria significantly contributes to unreliable prediction results [17]. The sampling procedure used [19] and the initial data preprocessing methods that lead to removal of outliers affect the prediction results [25]. Inappropriate tuning of model parameters [28] and the lack of robust benchmark models [9] also affect the reliability of SEE. Previous studies have proposed a simple baseline estimation model (automatically transformed linear model (ATLM)) based on a regression approach [9] and a complex estimation model (confidence guided effort estimator (CoGEE)) based on a bi-objective optimisation approach [6]. However, there are still discrepancies between these models, and it is difficult for software engineers to adapt them, due to the lack of guidance on the interpretation of the estimated results [19]. The state-of-the-art baseline model (ATLM) by Whigham et al. [9] only considers the estimation of effort and does not offer a corresponding interpretation guide for the estimated results. Irrespective of the SEE model, software researchers normally fail to consider the difficulties faced by software practitioners in comparing the results from different estimation models. This limits the practical usefulness of such estimation models [19,17]. Thus, current estimation models are trained from SEE datasets that contain independent variables (such as size, development duration, etc.) and a dependent variable (software effort), but do not consider the effort classes (see Fig. 1). Our contribution is a classification scheme that provides a self-guided interpretation of estimated effort (Fig. 2). The classification scheme is discussed in detail in Section 3.5.

To address the aforementioned issues faced by software engineers engaged in SEE, we consider the baseline model (ATLM) by Whigham et al. [9] and five other regression models for the empirical analysis. It should be noted that ATLM is a form of multiple regression model. The five regression models are ElasticNet, LASSO, Stepwise, Robust and Ridge regression, which are benchmarked against the baseline model (ATLM). It is worth noting that one of these regression models –

Proj_ID	KLOC	Duration	…	Effort	Class
1	449.9	13	…	336.3	?
2	289	14	…	116	?
3	254.2	13	…	258.7	?
4	214.4	18	…	86.9	?
5	40.5	7	…	82.5	?
….	…	…	…	…	…
n	200	11	…	130.3	?

Proj_ID	KLOC	Duration	…	Effort	Class
NewProj	60.2	14	…	???	???

Fig. 1. Effort estimation of projects without a classification scheme.

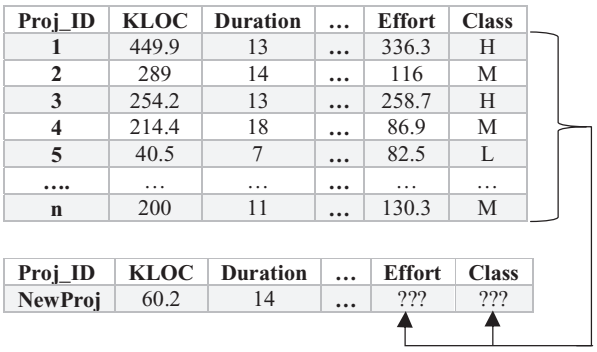

Fig. 2. Effort estimation of projects with a classification scheme. *H – high effort, M – moderate effort, L – low effort.*

ElasticNet – to the best of our knowledge, this is the first study to use it for SEE modelling. Current SEE studies [23,12,29] have considered regression models for predictive analysis. A previous study by Dejaeger et al. [14] confirmed that ordinary least squares regression (extended to build the ATLM) is the best prediction model out of a variety of SEE models.

This study makes three contributions.

1. The estimation of software effort in two dimensional outputs, namely the software development effort (*output 1*) and the software effort classification (*output 2*).
2. We introduce a scheme for classifying software effort to aid in the self-guided interpretation of software effort estimates for existing project data.
3. A comparative study of six different regression-based techniques, including the state-of-the-art baseline model (ATLM) and ElasticNet regression, with the objective of addressing the conclusion instability problem.

We find that ElasticNet regression, which is a hybrid model, can also form a baseline SEE model similar to ATLM. This implies that software engineers can use the ElasticNet regression to compare different prediction results from multiple SEE models. It should be noted that the regression models developed in this study have been subjected to rigorous statistical tests to confirm their efficacy in improving prediction accuracy.

The remaining sections of this paper are organised as follows. Section 2 reviews related work in SEE. Section 3 presents the research methodology and discusses the SEE datasets, preprocessing techniques, regression-based models, software effort classification scheme, proposed approach and evaluation measures. Section 4 presents the experimental results and discussion. Finally, Section 5 concludes the paper by highlighting the practical implications of the study's findings, threats to validity and future research directions.

2. Related work

Different techniques especially in the fields of machine learning, data mining and currently deep learning have been explored to aid in the construction of SEE models. Till date, none of these techniques can be referred to as the standard for SEE. To the best of our knowledge, this is the first study to consider the *duplex output SEE model* for the prediction of software effort as well as its respective classification for easy interpretation. As a result, our review section considers a number of recent studies that have considered SEE models to improve prediction accuracy.

A study by Menzies et al. [2] investigated if modern SEE models resulted in better prediction accuracy as compared to the traditional SEE models. Here, they compared the classical COCOMO SEE approach [30] to modern approaches such as the analogy based estimation [31] and the SEE baseline model (ATLM) by Whigham et al. [9]. The traditional Boehm's COCOMO model [30] was found to be more accurate for the COCOMO dataset as compared to the new modelling techniques.

Another study by Lokan and Mendes [23] investigated the concept of *moving windows* (use of recently completed projects as training set) based on linear regression model to improve the prediction accuracy in SEE. They found that the moving window modelling approach does not yield relative improved prediction accuracy as compared to the *growing portfolio* (use of all historical projects as training set). Thus, they reported that companies should not discard older projects but rather continue to use them for the training needs of a prediction model instead of using the moving windows. The Finnish dataset was used in the study. Evaluation of the prediction results from the regression modelling approach was done using the mean absolute error (MAE) and adjusted R^2. Similarly, linear regression was considered in previous SEE studies [11,29,32–35] exploring the moving window concept for improving prediction accuracy. These studies found that the moving window modelling improve prediction accuracy when different chronological datasets (e.g. ISBSG and Kitchenham) were used.

In our recent study [36], we considered a Deep learning approach (specifically Deep Neural Networks) for investigating the use of *prior* and *posterior features* for software testing effort estimation. *Prior features* refer to the available features before project completion whilst *posterior features* refer to features available after project completion. Here, a mixed integer linear programming (MILP) optimization was setup to select relevant projects with their respective features from within-company and cross-company projects. We found that the MILP optimization selected *prior features* which yielded similar prediction accuracy as compared to *posterior features* provided the data was normalised. This result was confirmed by the Friedman's test with *p-values* less than 0.05 asymptotic significance level. We found that when the training and validation sets were normalised, selection of relevant cross projects improved the prediction accuracy in modelling. Data normalisation resulted in an improved prediction accuracy in a previous study by Kocaguneli et al. [10] when finding essential features and project instances in SEE.

Keung et al. [7] investigated the *conclusion instability* issue with regards to inconsistent estimated results from different SEE datasets and multiple prediction models. In their study, 20 different SEE datasets, 10 data preprocessors and 9 prediction models (learners) evaluated on 7 performance measures were used. They realised that, aggregate analysis from multiple learners across multiple datasets assists in drawing stable conclusions in SEE. By empirical analysis and observation, Keung et al. [7] recommended that it is important for software engineers not to fit a single learner for a dataset in SEE.

Whigham et al. [9] researched into resolving the problem of assisting software engineers with a more accurate and robust SEE model. They proposed a baseline SEE model namely automatically transformed linear model (ATLM). This baseline model (ATLM) is simple and can be benchmarked with newly developed SEE models for easy adoption after it proved superior to a complex ensemble method and a hybrid approach incorporating particle swarm optimization, analogy-based estimation (ABE) and clustering technique. Even though multiple linear regression (MLR) model has not been considered as the best estimation model in other studies [37,38], Whigham et al. [9] contend that it can form a baseline model provided relevant data-driven transformations are applied. Their argument of using MLR as a proxy for the formulation of SEE baseline model was based on previous studies [14,39] and the following seven characteristics:

(1) Simple, easily implementable and easy to interpret output;
(2) Outcomes should be deterministic;
(3) No fine tuning of parameters during model construction;
(4) Relevant to a hybrid of quantitative and qualitative datasets;
(5) Output should be accurate than random guess;

(6) Provide relevant estimation information based on generalized dataset properties and
(7) Be made publicly available for easy implementation and execution.

Whigham et al. [9] argued that since there is no accepted standard for setting up a prediction model in SEE domain, their proposed ATLM can be used as a baseline model. ATLM supports the *n-fold* cross validation, the leave-one-out validation [18] and a suitable sample size with repeated training and validation sets. ATLM was benchmarked with the best ensemble method for effort estimation namely Artificial Neural Networks [3] and empirical analysis based on the 30-*fold* cross validation using three SEE datasets. Error evaluation measures namely MMRE, LSD, PRED(25) and Relative Error (RE*) were used to evaluate the prediction accuracy of the models. Even though no statistical test was considered in their study [9], results indicated that ATLM performed accurately better in estimating the software effort as compared to the complex ensemble method (ANN). Again, they compared ATLM to a hybrid approach incorporating particle swarm optimization, ABE and clustering as proposed by Bardsiri et al. [40]. They used two benchmark datasets – Cocomo81 and Maxwell, and concluded that ATLM performed accurately better over the hybrid approach [40]. Here, evaluation was done with the MMRE based on a 10-*fold* cross-validation approach.

A study by Dejaeger et al. [14] found that the ordinary least squares (OLS) regression model was the effective learner among a number of SEE learners used in estimating software effort. They used nine different datasets in their empirical analyses, namely coc81, cocnasa, desharnais, UPS05, Maxwell, ISBSG, ESA, experience and euroclear. Dejaeger et al. concluded that for a more accurate effort estimation model, there is the need for incorporating data transformation techniques, feature selection techniques, performance evaluation measures and statistical tests on a large training dataset. We have incorporated the recommendation of Dejaeger et al. [14] in the models developed in our study.

Turhan and Mendes [41] found that stepwise regression (SWR) performed better in the field of SEE specifically cross-company modelling. They applied the SWR model on 125 web projects from 8 companies with or without filtering technique. SWR is one of the modelling techniques being employed in this study since some of the data used are cross-company datasets.

Even though the aforementioned studies have contributed a lot in the domain of SEE, we argue that there is the need for classification of software effort to reduce the *conclusion instability* challenge [7,18]. The software classification methodology introduced in this study can estimate the level of effort expended irrespective of the numerous prediction models, evaluation measures, cross or within company projects and preprocessing techniques considered. Due to the fact that several SEE models have been proposed with no clear dominating technique, we build variations of regression models incorporating different preprocessing and feature subset selection techniques in assessing the performance on a number of different datasets. This study is based on the assessment made by Whigham et al. [9] and Dejaeger et al. [14] that a regression based model can form a simple and reliable prediction model for SEE. As a result of that, we incorporate six regression models to build *duplex output SEE models* for each of the 14 datasets investigated. Due to outliers, influential and missing data points in SEE datasets, we applied data transformation techniques as done in previous studies [10,29,14,23]. The uniqueness of this study is the classification of software effort in the studied datasets prior to modelling. This is to allow for self-guided interpretation of effort as well as enable comparison of the characteristics of the new project with past or completed projects. The extent of similarity between the new project and completed projects will hopefully increase the confidence in the effort estimation.

Table 1
Number of features and records in the datasets.

Dataset	Features	Size	Unit	Source
Albrecht	8	24	months	*GitHub*
China	17	499	hours	*PROMISE*
Cocomo81	17	63	months	*PROMISE*
Cocomonasa1	17	60	months	*PROMISE*
Cocomonasa2	24	93	months	*PROMISE*
Cosmic	11	42	months	*PROMISE*
Desharnais	11	81	hours	*PROMISE*
Kemerer	7	15	months	*PROMISE*
Kitchenham	7	145	months	*PROMISE*
Maxwell	26	62	hours	*PROMISE*
Miyazaki	8	48	months	*PROMISE*
php_projects	12	45	months	*Industry*
Telecom	3	18	months	*GitHub*
USP05	8	203	months	*GitHub*

3. Methodology

3.1. Datasets

We extracted 13 SEE datasets from *GitHub* (*github.com*) and the *PROMISE* repository [42] and another dataset from an industry partner (Table 1). The latter dataset contained 45 *PHP* projects from an industry partner whose identity cannot be disclosed due to confidentiality reasons. The details of all 14 datasets including the number of features, total number of instances (size) and software effort measurement units are presented in Table 1.

3.2. Data preprocessing techniques

To ameliorate data quality problems such as missing data, outliers and influential data points, the datasets were preprocessed in the following ways to ensure effective and efficient model construction.

Following Minku and Yao [3], we analysed all of the instances in each dataset to eliminate missing values. After a systematic analysis of the datasets, only a handful of missing entries (four instances) were observed in the Desharnais dataset, and no missing entries were found in the other datasets. To investigate the effect of outliers, we used the kernel density plots as recommended by Kitchenham et al. [43], residual plot analysis as recommended by Mendes and Kitchenham [44] and Cook's distance [45]. Following Mendes and Kitchenham's [44] approach to stability analysis, we investigated the effect of outliers in the studied datasets, as they can affect the regression analysis of the prediction performance. We first fit the regression models without considering data transformation techniques and investigated the effect of outliers. Secondly, we transformed the datasets, as done in previous studies, and investigated the effect of outliers in the transformed datasets. In each case, we evaluated the prediction performances of the regression models. The outliers were identified and removed if they added no significant effects to the construction of the prediction models [46,43,44,3]. We found that transforming the datasets prior to modelling, with or without outliers, resulted in better prediction accuracy than not transforming the data prior to modelling. We found that the outliers within the selected datasets did not significantly affect the models' prediction performances when the data were transformed. Following a similar approach by Seo and Bae [46], Cook's distance [45] was used in the identification and treatment of influential data points during the model construction. In this study, we found that the influential data points identified using Cook's distance [45] yielded no significant effects on the SEE modelling when the data transformation techniques were applied.

According to Lokan and Mendes [23], the error terms can be assumed to follow a normal distribution when setting up a regression-based model. We tested this normality assumption using the Shapiro-

Wilk test and Kolmogorov-Smirnov test [47].

We found that at a 5% asymptotic significance level, the sampled datasets without preprocessing were not normally distributed, due to data instability, data diversity, heterogeneity, variations and noise within the datasets [21,48,9,10]. According to Turhan [21], there is a need to address data instability within SEE datasets, as it affects the prediction accuracy of models. Therefore, we applied three data transformation techniques: *box-cox* [14], *z-score normalisation* [36,10] and log-*transformation* [23]. We checked for the normality assumption [23] in each transformed dataset and found that the residuals from both the *z-score* and log-*transform* followed the normal distribution. The normality was tested using both the Shapiro-Wilk and Kolmogorov-Smirnov tests [47]. The data transformation technique stabilised the variations and reduced the error terms in the highly skewed and heavy-tailed attributes, specifically the *Effort (response)* variable. We used a quantile-quantile (QQ) plot, a histogram and kernel density plots, as recommended by Kitchenham et al. [43], to observe the distribution of highly skewed attributes before and after the application of the transformation techniques. Upon assessment with the mean absolute error (MAE) evaluation measure [49], we found that the *z-score* normalisation yielded the best prediction accuracy for all of the datasets used in this study. We observed that the application of the *z-score* metric transformed the attributes within the datasets with a mean of 0 and variance of 1. As the *z-score* normalisation resulted in greater improvements in prediction accuracy than the log-*transform* and the *box-cox transform*, it was used to transform the data prior to building the *duplex output SEE model*.

Feature/attribute subset selection is a data reduction process that automatically searches for and selects the best subset of features (X_i) in a given dataset. It reduces overfitting, training time and improves the prediction accuracy of the estimation model [10,50]. In this study, we adopted the wrapper approach, which evaluates attribute subsets based on a number of selected learning algorithms. The following five feature selection algorithms were applied to the training set: *Best First (BF), Subset Size Forward Selection (SSFS), Genetic Search Algorithm (GSA), Linear Forward Selection (LFS)* and *Random Search (RS)*. The most suitable algorithm was selected based on the minimum MAE, minimum BMMRE and maximum adjusted R^2. These feature subset selection algorithms are discussed in Section 3.3.

In this study, we used the leave-one-out (LOO) validation approach to meet the training and validation needs of model construction. The LOO approach was used because it 'removes the cause of *conclusion instability*' in assessing SEE models, as noted by Kocaguneli and Menzies [18]. Specifically, LOO helps to overcome the problem of the random selection of training and validation sets in the setup of the effort estimation model. Thus, for each dataset of size ***N***, we iteratively used ***N-1*** instances for training and the remaining project for validation. Each model's performance was assessed using the three evaluation measures at each iteration phase to obtain the best training and validation sets.

3.3. Feature selection techniques

3.3.1. Genetic search algorithm (GSA)

The genetic search algorithm (GSA) is an optimization method which mimics a biological evolution principle of solving a constrained or unconstrained search space problem [51]. GSA initially generates a random population set. It is an iterative procedure consisting of the following processes: 1) objective function evaluation, 2) reproduction of population set in order to obtain feasible solution, 3) crossover whereby existing (parent) populations are mated in order to produce new (offspring) population and 4) mutation whereby random changes are made on a subset of the population. This approach has been considered for feature subset selection problems by Andrews and Menzies [52].

3.3.2. Best-first (BF)

The best-first (BF) algorithm finds the optimal feature subset without necessarily evaluating every node according to a specified rule [53]. The algorithm makes use of a lower-bound function to determine which nodes to prune off in the feature search space. BF can start at any point in the search space based on the backward and/or forward search process with the aim of pruning off irrelevant nodes (or features) and maintaining the best nodes. BF algorithm has been used by Morrison et al. [54] for the search space selection problem.

3.3.3. Subset size forward selection (SSFS)

The subset size forward selection (SSFS) algorithm performs search process based on an interior *k-fold* cross validation to find the optimal subset-size of features for modelling [55]. A subset size evaluator [55] is performed on the whole dataset to obtain the optimal feature subset. This approach is incorporated in the selection of the best feature subset based on a reduction of the number of wrapper evaluations [56].

3.3.4. Linear forward selection (LFS)

The linear forward selection (LFS) algorithm is performed by taking a restricted initial number of features. In each iteration of the selection process, the number of features increases [55]. LFS algorithm uses an ordering or ranking measure to obtain the best features for the modelling process. This approach for feature subset selection problem has been considered by Bermejo et al. [56].

3.3.5. Random search (RS)

The random search (RS) algorithm performs a random search starting from a random or specified point in the feature space and records the best feature subset obtained [57]. This search process is enhanced with the Las Vegas filter algorithm [57] which improves the efficiency of the selection of the best subset from the feature space. This approach has been used for feature subset selection problems by Bertolazzi et al. [58].

3.4. Regression based effort estimation techniques

The following six regression based techniques were setup for the purpose of this study and evaluated based on three performance evaluation measures as discussed in Section 3.7.

3.4.1. Ordinary least squares regression (OLS)

OLS also referred to as linear regression is one of the most popular and widely used SEE models [14,29,12,23]. OLS was considered optimal in an effort estimation study by Dejaeger et al. [14]. Whigham et al. [9] built on OLS to propose the ATLM, state-of-the-art baseline SEE model. With regards to ATLM [9] and the OLS by Dejaeger et al. [14], the dependent variable is continuous whilst the independent variable(s) can be continuous or discrete. In this study, we use a continuous dependent variable (effort) in addition to multiple independent variables for the regression analysis. Hence, the multiple OLS used in this study is defined in (1) as follows:

$$Effort_i = \beta_0 + \sum_{j=1}^{p} \beta_i x_{ii} + \varepsilon_i \quad (1)$$

where β_0 and β_i are the regression coefficients, x_{ij} is the j^{th} independent (predictor) variable of the i^{th} project out of a total p independent variables. $Effort_i$ is the dependent (response) variable of the i^{th} project.

3.4.2. Stepwise regression (SWR)

This type of regression is used when there are multiple independent (predictor) variables. Here, the selection of predictor variables for the model was done based on an automatic addition and deletion process. We used both the forward and backward variable selection techniques based on the order of importance of the variables [59]. The aim is to use

minimum predictors to maximize the prediction power. The stepwise regression (SWR) model is different from OLS based on the forward and backward variable selection techniques.

3.4.3. Ridge regression (RR)

Unlike SWR which restricts the number of predictor variables in the model, ridge regression (RR) shrinks the regression coefficients of highly correlated predictors close to zero. This is done by introducing the ridge parameter, α [60] resulting in the regression coefficients, β as defined in (2).

$$\hat{\beta} = (X'X + \alpha I_n)^{-1}(X'\varepsilon) \tag{2}$$

where $X'X$ is the singular matrix of predictors, I_n is the identity matrix of rank n and ε are the error terms.

3.4.4. LASSO regression (LASSOR)

Similar to RR, least absolute shrinkage and selection operator (LASSO) also penalizes the number of predictor variables entering the model using the L_1 regularisation method. Here, if group of predictors are highly correlated, the technique picks only one group and shrinks the others to zero. LASSOR estimates the regression coefficients, β as defined by Tibshirani [61] in (3).

$$\begin{aligned}\hat{\beta} &= \arg_{\beta\in\mathbb{R}^p}\min \|y - X\beta\|_2^2 + \lambda\sum_{j=1}^{p}|\beta_j| \\ &= \arg_{\beta\in\mathbb{R}^p} \|y - X\beta\|_2^2 + \lambda \|\beta\|_1\end{aligned} \tag{3}$$

LASSOR is different from RR since it uses L_1 penalty $\|\beta\|_1$ and RR uses L_2 penalty$\|\beta\|_2^2$ yielding different estimates [61].

3.4.5. ElasticNet regression (ENR)

This technique proposed by Zou and Hastie [62] is a regularisation and feature selection technique which helps in eliminating highly correlated predictor variables from the model. It is a hybrid of RR and LASSOR. This is useful when there are multiple predictors which are highly correlated. ENR finds the optimal β by solving the following problem as defined by Friedman et al. [63].

$$\begin{aligned}\hat{\beta} &= \arg_{b\in\mathbb{R}^{p+1}}\min\left[\frac{1}{2n}\sum_{i=1}^{n}(y_i - x_i^T b)^2 + \lambda P_\alpha(b_1,\cdots,b_p)\right] \text{where } P_\alpha \\ &= \sum_{i=1}^{p}\left[\frac{1}{2}(1-\alpha)b_j^2 + \alpha|b_j|\right]\end{aligned} \tag{4}$$

P_α is the elastic-net penalty [62] which is useful in solving the highly correlated predictor variables and also works best when the number of observations (n) is less than the number of predictors (p) for $\alpha \in (0,1)$ [63]. RR and LASSOR are obtained from ENR model in Eq. (4) for $\alpha = 0$ and $\alpha = 1$ respectively [63].

To the best of our knowledge this is the first study in software effort estimation to employ the ElasticNet regression which has been successfully used in other domains such as the biomedical field [64] and image processing [65].

3.4.6. Robust regression (RoR)

Robust regression (RoR) is an alternative to OLS regression and more efficient when datasets are contaminated with outliers or influential observations [66,67]. RoR minimizes the objective function in (5) whereby the weights, w_i are iteratively adjusted by taking the error terms, ε_{i-1} of the previous iteration into consideration.

$$\min\sum_{i=1}^{n} w_i^2\varepsilon_i^2 \tag{5}$$

3.5. Software effort classification scheme

To provide self-guidance in the interpretation of the level of effort expended, we introduced a classification scheme for SEE. Software effort classification is the process of categorising the estimated software effort into classes. The classification system was based on historical project datasets (historical data being the same as a training set in this study). A goal of the software effort classification scheme was to facilitate the interpretation of the estimated software effort (Y_R) in the context of existing organisation (training) data. The classification scheme had three classes – *low, moderate* and *high*. To define these levels of classification, we first discretised the actual effort levels recorded in the historical SEE datasets into three classes based on the density quantile theory introduced by Eubank [26]. The density quantile theory was used because it produces optimal spacing selection threshold values for categorising effort into classes. The density quantile function, $Q(u)$ is defined in (6).

$$Q(u) = F^{-1}(u),\ 0 \le u \le 1, \tag{6}$$

where F is the continuous distribution function of a random sample, $X_1, \ldots, X_n$ and u is the location parameter for the optimal spacing selection threshold [26]. As a result of the data normalisation technique applied to the SEE datasets, the actual and continuous effort values were scaled to fall within a specific normalisation range with a mean of 0 and a standard deviation of 1. The thresholds for the effort classifications of each of the preprocessed and normalised SEE datasets were computed using the density quantile function [26].

3.6. Duplex output algorithm

The *duplex output SEE model* is constructed based on Algorithm 1. The input comprises of training and validation sets with predictor variables - ***features(X)***, the response variable - ***effort(Y_1)*** with its

Algorithm 1
Duplex Output Model (X_r, Y_R, Y_Q).

Input: Features (X), Effort (Y_1) and Classification (Y_2)
Outputs: Output (Y_R) from regression model and Classification Output (Y_Q)
$N \leftarrow$ *total number of dataset instances*
$p \leftarrow$ *total number of features*
$r \leftarrow$ *subset of p features*
$\beta_r \leftarrow$ *Regression Coefficients*
$Q_l \leftarrow$ *Threshold for Low Effort class*
$Qm \leftarrow$ *Threshold for Moderate Effort class*
$Q_h \leftarrow$ *Threshold for High Effort class*
Begin
Load project dataset of size N
Preprocess dataset
// Attribute Selection using GSA
for *i, i = 1,…,p* ***do***
  select X[r] ← X[1] to X[i]
end for
Apply normalisation technique
for *j, j = 1,…,N* ***do***
  Perform LOO validation
end for
for *k, k = 1,…,r* ***do***
  Compute $\beta_r \leftarrow$ *inv(X'X)X'Y*
end for
Test for β_r *using AIC and Welch t-test*
Select best β_r *for model construction based on VIF*
// Construct Regression model
$Y_1 \leftarrow \beta_0 + \Sigma\beta_i X_i$
Output(Y_R) // Estimated Effort
// Effort classification
Effort classification based on discretization function
If *Output(Y_R)* $< Q_l$ ***then***
  Output(Y_R) as "Low Effort"
  else if *Output(Y_R)* $\ge Q_h$ ***then***
    Output(Y_R) as "High Effort"
  else *Output(Y_R) as "Moderate Effort"*
end if
Output(Y_Q) // Effort classification
End

respective level of classification variable - ***classification(Y_2)***. We first load each dataset with ***N*** number of instances in step 1 and apply preprocessing techniques to resolve all forms of variations in the dataset (step 2) as elaborated in Section 3.2. In steps 3 to 5, a subset of ***r*** features are selected from a total of ***p*** features. Here, the input feature subset with the minimum MAE, minimum BMMRE and maximum adjusted R^2 is used in constructing the *duplex output model*. Thus, per our empirical analysis, GSA proved more efficient over the 4 feature subset selection techniques with respect to prediction accuracy. The feature subset is subjected to the *z-score* normalisation technique (step 6). We apply the leave-one-out (LOO) cross validation approach on the datasets and assessed each partition set performance (steps 7–9) with the three evaluation measures. The partition set with the best accuracy performance was considered for constructing the SEE model. The regression coefficients, $\boldsymbol{\beta_r}$ were computed in steps 10–12 for each of the regression models. In step 13, the $\boldsymbol{\beta_r}$ values were tested using Welch's *t*-test statistic as recommended by Kitchenham et al. [43]. This test is effective for unequal data samples assumed to have inconsistencies or unequal variances which are statistically different from each other. The number of predictor variables can affect the model performance, hence preprocessing techniques and multicollinearity diagnostic tests were used in order to obtain the best feature subset for the regression models. Akaike information criterion (AIC) and variance inflation factor (VIF) were used to check for multi-collinearity among the ***Xr*** to obtain the best ***βr*** (step 14). We considered multicollinearity as an issue if the VIF is more than a specified threshold of ten [68] in each AIC model. The multicollinearity diagnostic analysis was confirmed based on the use of the *t*-test statistic [68]. The $\boldsymbol{\beta_r}$ values were used to construct the regression model in step 15 and the estimated effort values are denoted as ***Output(Y_R)*** in step 16. The ***Output(Y_R)*** of the regression model is then classified into its respective class (step 17). The estimated effort, ***Output (Y_R)*** was categorised into their respective effort classes, namely *low, moderate* and *high* efforts (steps 18–23). This is done based on the density quantile function [26] thresholds to classify the effort values into low (Q_l) and high (Q_h) classes respectively (step 18–23). Thus, estimated effort values less than Q_l are classified as *Low Effort* and estimated effort values more than Q_h are classified as *High Effort*. Lastly, estimated effort values falling within the thresholds [Q_l Q_h] are classified as *Moderate Effort*. The algorithm ends with the estimated effort classification as the second and auxiliary variable *Output(Y_Q)* (step 24).

3.7. Performance evaluation measures

In order to check the performance of the developed models, three evaluation criteria were used, namely mean absolute error (MAE), balanced mean magnitude of relative error (BMMRE) and adjusted R^2. MAE is a good indicator for evaluating the average performance of a given prediction model as noted by Shepperd and MacDonell [24], Kocaguneli et al. [10], Sarro et al. [6], Willmott and Matsuura [69], and Borji et al. [70]. BMMRE is considered over MMRE due to MMRE ratio biasness, unbalanced measure, overestimation and other criticisms as noted by Foss et al. [49] and Kumar and Yadav [71]. Adjusted R^2 provides a more accurate judgement of the goodness of fit of an estimation model as reported by Lokan and Mendes [23,32].

3.7.1. Mean absolute error (MAE)

MAE is a risk function that measures on average how the estimated effort (E_E) values deviate from the actual or true effort (E_A) values and it is defined in (7). Note that n is the sample size of the data under consideration.

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |E_{Ai} - E_{Ei}| \tag{7}$$

3.7.2. Balanced mean magnitude of relative error (BMMRE)

In order to alleviate the unbalanced and the inability to overestimate, Miyazaki et al. [67] proposed the BMMRE metric defined in (8). BMMRE was proven reliable in the selection of the best prediction model by Kumar and Yadav [71].

$$BMMRE = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{|E_{Ai} - E_{Ei}|}{\min(E_{Ai}, E_{Ei})}\right) \tag{8}$$

3.7.3. Adjusted R^2

Adjusted R^2 defined in (9) is used to measure the goodness of fit which penalizes and optimizes the number of input features or independent variables to be included in each regression model [23]. This metric helps to estimate the total variation in the response variable (effort values) which has been accounted for by the independent variables.

$$Adjusted\ R^2 = 1 - \frac{(1 - R^2)(n - 1)}{n - p - 1} \tag{9}$$

where p is the number of predictor variables and R^2 is the multiple coefficient of determination.

Thus, for a model to achieve a relative prediction accuracy, it must have a minimum MAE, minimum BMMRE and maximum adjusted R^2 respectively.

3.8. Experimental setup

We conducted an empirical analysis to answer three research questions. Each SEE model was set up using the leave-one-out (LOO) cross validation approach [18]. We evaluated the models' prediction accuracy using MAE, BMMRE and adjusted R^2.

RQ1. *Does feature subset selection improve the accuracy of the multiple regression models?*

We applied five feature subset selection techniques, namely BF, SSFS, GSA, LFS and RS (see Section 3.3), to each of the datasets. The intention was to assess the prediction accuracy of each of the wrapper approaches (feature subset algorithm + regression-based model) and obtain the best feature subset algorithm, which was then incorporated into the multiple regression model. We applied the wrapper approach because Chen et al. [50] showed that a model's predictive power can be improved through the use of feature selection. Similarly, Kocaguneli et al. [10] concluded that the essential content of an effort estimation dataset can be characterised by a subset of features used in the modelling. The MAE, BMMRE and adjusted R^2 evaluation measures were used to assess the performance accuracy of the different feature subset selection techniques. Thus, in each wrapper approach, the respective feature subset technique together with each of the six regression techniques were used and the prediction accuracy was assessed by averaging the MAE, BMMRE and adjusted R^2 for each dataset. A test for multicollinearity based on the variance inflation factor (VIF) and the stepwise procedure [23,32] was used to assess the inclusion and exclusion criteria of the feature subsets. Again, at the 5% asymptotic significance level, the Kruskal-Wallis *H-test* [72] was applied to test the significance of the inclusion of the feature subsets from each dataset. In each case of model construction, the Akaike information criterion (AIC) backward selection procedure was incorporated into each regression analysis to further confirm the inclusion of the input feature subset. It should be noted that the LOO validation [18] was used to partition the dataset into training and validation sets. This was repeated until each partition was used as part of the training and validation set.

To confirm that the feature subset selection process improved the models' accuracy performance, we performed another empirical analysis in which we constructed SEE models using all of the features in each dataset.

RQ2. *Can the level of software effort estimates be correctly classified?*

After obtaining the best feature subset selection algorithm based on the performance evaluation metrics and the aforementioned statistical tests, the estimated effort of each dataset was classified. The intention was to determine whether the effort levels were correctly classified across the multiple regression models. This was done as follows: for each dataset, we applied the density quantile function [26] to discretise the effort values into three classes. Thus, actual effort values below the lower quartile (Q_l) formed the low effort class and values above the upper quartile (Q_h) formed the high effort class, with the values between Q_l and Q_h forming the moderate effort class. We constructed each multiple regression algorithm to estimate the software effort (first output). Then, we assigned each estimated effort into a class (second output) based on the thresholds identified by the density quantile function. Estimated effort values that fall outside the thresholds of the three classes can be an indication to software engineers to pay special attention to such projects, as they might have characteristics that are new to the organisation.

We evaluated the accuracy of the estimated effort classification produced by the multiple regression models using MAE, BMMRE and adjusted R^2 evaluation measures. Thus, minimum MAE, minimum BMMRE and maximum adjusted R^2 values indicated that the *duplex output model* had superior prediction accuracy. Following a similar approach by Kocaguneli et al. [8] and Mensah et al. [36], each model was set up using the LOO validation [18] and the number of prediction losses were recorded.

We further performed a statistical test using the Welch's *t*-test statistic [43,73] to confirm the classification results for the effort estimates at 1% and 5% asymptotic significance levels. This statistical test was performed to investigate whether there were significant differences between the classes of the actual effort values and the classes of the estimated effort values for each multiple regression model. Welch's *t*-test, a robust statistical test [43], was chosen as it can test for heterogeneous (unequal) variances within datasets. Thus, as the three estimated effort classes were assumed to have heterogeneous variances, it was necessary to use a more robust test statistic to test the accuracy of the classifications of estimated effort produced by the multiple regression analyses.

RQ3. *How is the accuracy of the regression-based models compared to the accuracy of the state-of-the-art baseline model (ATLM)?*

We considered the five regression-based models and performed an empirical analysis to benchmark the results against the state-of-the-art baseline model, ATLM [9]. The aim was to investigate the existence of significant differences in performance accuracy between the regression-based models. The GSA was used to incorporate the subset selection of features and the models were set up with the LOO validation and evaluated using MAE, BMMRE and adjusted R^2. We first used the Kruskal-Wallis *H*-test statistic to test the null hypothesis that '*there is no statistically significant difference in accuracy across the regression models*' [72]. We chose this test statistic because of its robustness to outliers. It proved effective in a study by Krishna et al. [74] and Mensah et al. [35] that compared the effectiveness of *Bellwether* projects. Secondly, following a similar approach by Whigham et al. [9], we compared the regression models with the state-of-the-art model. The results for the three evaluation measures, MAE, BMMRE and adjusted R^2 across the models are presented in Section 4. For a model to be assessed as having relative prediction accuracy, it had to have a minimum MAE, minimum BMMRE and maximum adjusted R^2.

4. Results and discussion

4.1. RQ1. Does feature subset selection improve the accuracy of the multiple regression models?

This study applied five feature selection techniques. Below, we present the recorded MAE, BMMRE and adjusted R^2 evaluation measures for each of the five selection techniques. The average MAE, BMMRE and adjusted R^2 measures were recorded for each dataset using the six regression models and each of the five feature subset selection algorithms (Tables 2 – 4, respectively). The results presented in Tables 2, 3 and 4 indicate that, on average, the genetic search algorithm (GSA) had the minimum MAE value (20.4%), minimum BMMRE value (21.5%) and maximum adjusted R^2 value (68.6%) across all of the normalised datasets and all of the regression models. This is consistent with a study by Andrews and Menzies [52] that found GSA to be the

Table 2
Evaluation of feature subset selection algorithms based on MAE (%).

Dataset	BF	SSFS	GSA	LFS	RS	None
Albrecht	21.1	14.5	14.1	21.2	34.2	21.2
China	20.2	20.2	11.6	20.2	35.3	20.9
Cocomo81	27.1	27.1	15.7	27.1	16.9	45.8
Coconasav1	35.1	35.1	17.3	35.1	19.5	47.0
Coconasa2	33.5	33.5	19.3	33.5	21.1	45.9
Cosmic	49.6	49.6	34.5	49.6	51.2	51.2
Desharnais	28.8	29.8	12.4	29.8	33.2	43.5
Kemerer	33.6	41.1	33.6	33.6	35.6	65.5
Kitchenham	8.8	8.8	8.8	8.8	8.8	9.2
Maxwell	33.7	45.9	17.5	33.7	46.8	46.8
Miyazaki	22.4	22.4	22.4	22.4	22.4	21.1
php_projects	15.7	15.0	15.0	15.0	15.0	17.2
telecom	39.1	39.1	39.1	39.1	39.1	45.6
USP05	23.8	23.8	23.8	23.8	27.6	29.0
Mean	**28.0**	**29.0**	**20.4**	**28.1**	**29.1**	**36.4**

Note: 'None' indicates that no feature subset selection algorithm was considered during the modelling.

Table 3
Evaluation of feature subset selection algorithms based on BMMRE (%).

Dataset	BF	SSFS	GSA	LFS	RS	None
Albrecht	35.5	27.2	13.2	24.7	34.9	35.5
China	20.9	25	10.6	31.9	34.1	20.9
Cocomo81	34.2	27.9	14.6	15.9	18.9	34.2
Coconasav1	57.8	56.8	53.4	50.2	56.7	57.8
Coconasa2	38.7	40.1	32.6	37.8	43.1	38.7
Cosmic	28	24.5	22.9	28.1	27.3	28
Desharnais	18.6	19.6	14.5	23.4	19.2	18.6
Kemerer	17.9	20.8	19.1	20.7	18.8	17.9
Kitchenham	16.2	17.4	11.3	12.7	17.3	16.2
Maxwell	39.6	46.6	27.5	49.9	41.2	39.6
Miyazaki	23.8	43.5	19.3	33.5	31.1	23.8
php_projects	39.6	18.6	24.5	26.1	26.7	39.6
telecom	20.9	22.5	12.4	18.4	33.8	20.9
USP05	27.4	22.3	24.5	23.7	23.7	27.4
Mean	**29.9**	**29.5**	**21.5**	**28.4**	**30.5**	**29.9**

Table 4
Evaluation of feature subset selection algorithms based on adjusted R^2.

Dataset	BF	SSFS	GSA	LFS	RS	None
Albrecht	0.44	0.57	0.71	0.52	0.48	0.44
China	0.60	0.28	0.78	0.45	0.17	0.60
Cocomo81	0.71	0.58	0.70	0.56	0.37	0.71
Coconasav1	0.38	0.39	0.41	0.41	0.32	0.38
Coconasa2	0.39	0.42	0.57	0.40	0.52	0.39
Cosmic	0.61	0.61	0.66	0.57	0.50	0.61
Desharnais	0.64	0.64	0.77	0.59	0.60	0.64
Kemerer	0.58	0.55	0.62	0.49	0.48	0.58
Kitchenham	0.76	0.76	0.79	0.76	0.72	0.76
Maxwell	0.57	0.57	0.75	0.75	0.61	0.57
Miyazaki	0.64	0.58	0.65	0.62	0.55	0.64
php_projects	0.67	0.70	0.77	0.67	0.69	0.67
telecom	0.78	0.73	0.79	0.67	0.70	0.78
USP05	0.60	0.54	0.65	0.65	0.62	0.60
Mean	**0.60**	**0.57**	**0.69**	**0.58**	**0.52**	**0.60**

Note: Adjusted R^2 lies in the range [0–1].

Table 5
Optimal number of features selected by GSA.

Dataset	Input features	
	Total	Optimal
Albrecht	7	3
China	16	10
Cocomo81	16	4
Cocomonasa1	16	7
Cocomonasa2	23	7
Cosmic	10	5
Desharnais	10	5
Kemerer	6	3
Kitchenham	6	3
Maxwell	25	19
Miyazaki	7	2
php_projects	11	5
telecom	2	1
USP05	7	4

Table 7
Software effort classification based on density quantile function.

Dataset	Software effort classification thresholds		
	Low	Moderate	High
Albrecht	0–0.32	0.33–0.64	≥ 0.65
China	0–0.24	0.25–0.54	≥ 0.55
Cocomo81	0–0.25	0.26–0.37	≥ 0.38
Coconasav1	0–0.29	0.30– 0.57	≥ 0.58
Coconasa2	0–0.23	0.24–0.52	≥ 0.53
Cosmic	0–0.21	0.22–0.56	≥ 0.56
Desharnais	0–0.31	0.32–0.57	≥ 0.57
Kemerer	0–0.24	0.25–0.56	≥ 0.57
Kitchenham	0–0.13	0.14–0.26	≥ 0.27
Maxwell	0–0.22	0.23–0.65	≥ 0.66
Miyazaki	0–0.16	0.17–0.30	≥ 0.31
php_projects	0–0.24	0.25–0.60	≥ 0.61
Telecom	0–0.26	0.27–0.69	≥ 0.70
USP05	0–0.22	0.23–0.31	≥ 0.32

best and fastest feature subset algorithm for setting up a model. Table 2 shows that when no feature selection algorithm was used in the model, the average performance evaluation based on MAE across the datasets was 36.4%. This maximum MAE value for the non-application of feature subset selection algorithms shows the need to prune irrelevant features prior to modelling. Similar results were obtained by the BMMRE and adjusted R^2 evaluation measure, shown in Tables 3 and 4, respectively. This is consistent with a study by Kocaguneli et al. [10] that found that the essential content of an effort estimation dataset can be characterised by a subset of features and that using the subset improves prediction accuracy [52].

The optimal number of selected features for the different datasets obtained by the GSA are presented in Table 5. These values were used to construct the multiple regression-based models. The analysis of the multicollinearity test showed that the VIF indicated that none of the input features selected by the GSA should be excluded, as their respective VIFs were less than the threshold of 10 stipulated by O' Brien [68]. The average *F*-test statistic and the *p-values*, multiple correlation coefficients (R), adjusted R^2 and standard errors (SE) of the estimates were recorded for all six regression models for each dataset (Table 6). The results indicated that at the 5% significance level, the selected input features were suitable for use in the multiple regression models, as each *p*-value was less than the 0.05 significance level. The multiple correlation coefficients (R) for all of the datasets were within the range of 0.55–0.98 indicating a generally moderate to strong relationship between the input features and the effort response variable. Similarly, the adjusted R^2 values were on average within the range of 0.40–0.97 indicating that about 40 - 97% of the total variation in the effort response variable was accounted for by the selected input features. Again, at the 1% and 5% significance levels, the Welch's *t*-test statistic for the significance of each input feature in the *duplex output model* confirmed the validity of the input attribute subset selection of each dataset. The standard errors (SE) between the estimated efforts and the actual efforts for all five models fell, on average, in the 0.14–0.75 range (Table 6), indicating that the selected features for the *duplex output modelling* improved accuracy when the datasets were preprocessed and normalised. It should be noted that in each case of model construction, the AIC backward selection procedure was incorporated to further confirm the inclusion of the input subset selection of the features. Hence, we considered GSA the best feature subset selection algorithm for the multiple regression models when the datasets were normalised.

Table 6
Statistical diagnostic tests.

Dataset	*F*-test	*p*-value	R	Adj. R^2	SE
Albrecht	129.30	.004**	.975	.944	.175
China	40.49	.004**	.673	.442	.617
Cocomo81	9.63	.000**	.632	.458	.697
Coconasav1	33.89	.000**	.906	.796	.344
Coconasa2	5.22	.003**	.578	.401	.717
Cosmic	5.46	.022*	.552	.399	.747
Desharnais	17.59	.001**	.693	.483	.513
Kemerer	5.08	.017*	.819	.638	.542
Kitchenham	4853.5	.001**	.987	.973	.156
Maxwell	16.51	.000**	.861	.696	.442
Miyazaki	524.19	.000**	.970	.964	.137
php_projects	232.62	.005**	.972	.940	.188
telecom	11.93	.003**	.654	.427	.543
USP05	52.60	.000**	.597	.405	.748

Significance: *p < .05, **p < .01.

4.2. RQ2. Can the level of software effort estimates be correctly classified?

Table 7 shows the software effort classification thresholds for each of the normalised datasets based on the use of the density quantile function [26]. The low and high effort class thresholds were discretised at the 25% and 75% quartiles, and the moderate effort class range was between the 25%–75% quartiles. Given the high rate of inconsistencies within effort estimation datasets, we agreed with other researchers [28,75,25] that to accurately predict software effort, the datasets must be subjected to a normalisation technique. This is a necessary step in the correct classification of the estimated effort.

To confirm the above hypothesis, we classified the estimated effort values in each dataset using each of the six multiple regression models and evaluated their classification performance using MAE, BMMRE and adjusted R^2. Table 8 presents the recorded losses of the estimated effort classifications for each dataset with respect to MAE performance evaluation. As we used the LOO validation approach, we recorded the number of losses for each project's estimated effort class per a dataset. Thus, a loss was recorded when the estimated effort class was different from the actual effort class for at least one project for each SEE dataset. To be able to assess the accuracy of the classification predictions of the multiple regression models, the effort classes must be the same, despite the variations between the effort estimates of the different prediction models. In this analysis, a loss denoted by a '1' indicates a wrong classification of the software effort by the prediction model. The results show that the ElasticNet regression (ENR) had the minimum number of losses (i.e. two losses) in the estimation of the effort classification (Table 8). This was followed by the state-of-the-art baseline regression model, ATLM [9], and the LASSO regression which had a minimum of four losses. Thus, the ENR model can more accurately predict effort classes than the baseline model (ATLM) and the other regression models examined in this study. The results from the LASSO regression yielded

Table 8
Classification performances of the estimated effort w.r.t. number of losses.

Dataset	ATLM	RR	LASSO	ENR	RoR	SWR
Albrecht	1	1	0	0	1	0
China	0	1	0	1	1	1
Cocomo81	0	1	1	0	1	0
Coconasav1	0	0	0	0	0	1
Coconasa2	0	0	0	0	1	1
Cosmic	1	1	1	1	1	1
Desharnais	0	1	1	0	0	1
Kemerer	1	1	0	0	1	0
Kitchenham	0	0	0	0	0	0
Maxwell	0	0	0	0	0	0
Miyazaki	0	0	0	0	0	0
php_projects	1	1	0	0	1	0
telecom	0	0	0	0	1	0
USP05	0	0	1	0	1	1
Sum of losses	**4**	**7**	**4**	**2**	**9**	**6**

Note: '1' denotes a LOSS when the estimated effort classification is different from the actual effort, whereas '0' denotes a WIN, implying a correct classification.

fewer losses than the SWR, confirming the results of a previous study by Tibshirani [61]. Similar results were obtained for the BMMRE and adjusted R^2 evaluation metrics. For brevity, we present only the MAE results. The classes of the mean estimated efforts for the 14 SEE datasets calculated by the six *duplex output models* are provided in Table 9. The average classes (*high (H), moderate (M)* and *low (L)*) of the actual efforts for each SEE dataset are also presented in Table 9.

We validated the estimated effort classification from the six multiple regression models using Welch's *t*-test (Table 10). We also tested the null hypothesis that '*there is no statistically significant difference between the estimated and actual effort classification made by the duplex output models without assuming equality of variances*' . The Welch's *t*-test results (depicted in Table 10) show that there were no significant differences between the estimated effort classifications and the actual effort classifications for each of the six models across the datasets. Thus, the *p*-values for each *duplex output model* were all greater than the 0.05 and 0.01 asymptotic significance levels. Thus, irrespective of the unequal variances within datasets, the Welch's *t*-test results confirmed that the level of prediction accuracy of the estimated effort classification was within the 95% and 99% confidence intervals. It should be noted that the degrees of freedom of the Welch's *t*-test are decimals, as the test statistic is comprised of variances that are usually non-integers [43].

Table 10
Welch's *t*-test for software effort classification.

Duplex model	Welch's *t*-test		
	t-value	*df*	*p*-value
ATLM	-0.6529	25.03	0.5198
RR	-1.8688	25.79	0.0730
LASSOR	-0.7555	24.01	0.4573
ENR	0.3392	23.96	0.7374
RoR	-0.9833	24.93	0.3349
SWR	-0.9464	13.00	0.3612

Significance: *p < .05, **p < .01.

Table 11
Kruskal-Wallis *H*-test for model evaluation.

Evaluation metric	*Chi-square*	*p*-value
MAE	6.63	0.024*
BMMRE	7.73	0.011*
Adjusted R^2	10.09	0.028*

Significance: *p < .05, **p < .01.

4.3. RQ3. How is the accuracy of regression-based models compared to the accuracy of the state-of-the-art baseline model (ATLM)?

Table 11 shows the results of the Kruskal-Wallis *H*-test for the null hypothesis that '*there is no significant difference between the accuracy of the regression models*' . We present the Chi-square and *p*-values for the comparison of the MAE, BMMRE and adjusted R^2 values of the six regression models in Table 11. The results show that the *p*-values for the three evaluation metrics were all significant at the 5% asymptotic significance level, indicating that the null hypothesis can be rejected. We therefore concluded that there are significant differences in the models' prediction performance when all three evaluation metrics were used in the assessment.

We present the results of the performance evaluation measures for all the six models applied to all 14 datasets in Tables 12, 13 and 14, which show the values for MAE, BMMRE and adjusted R^2, respectively. For all three evaluation measures, we did not get the same performance evaluation for all six models across the datasets. We therefore assessed the performance of the six regression models based on the sum of the models' performance in all 14 datasets. The greyed cells indicate the best values and we summed the greyed cells for easy comparison and

Table 9
Actual vs. estimated effort values and their classification.

Dataset	Normalised effort (Level of class)						
	Actual	ATLM	RR	LASSO	ENR	RoR	SWR
Albrecht	0.66 (H)	0.61 (M)	0.48 (M)	0.68 (H)	0.74 (H)	0.60 (M)	0.82 (H)
China	0.56 (M)	0.60 (H)	0.47 (M)	0.57 (H)	0.57 (H)	0.37 (M)	1.24 (H)
Cocomo81	0.49 (H)	0.39 (H)	0.35 (M)	0.08 (L)	0.61 (H)	0.35 (M)	0.68 (H)
Coconasav1	0.64 (H)	0.83 (H)	0.65 (H)	1.25 (H)	1.06 (H)	0.70 (H)	0.57 (M)
Coconasa2	0.56 (H)	0.64 (H)	0.57 (H)	1.13 (H)	1.05 (H)	0.49 (M)	0.08 (L)
Cosmic	0.59 (H)	0.52 (M)	0.02 (L)	0.08 (L)	0.56 (M)	0.39 (M)	0.16 (L)
Desharnais	0.72 (H)	0.87 (H)	0.56 (M)	0.19 (L)	0.69 (H)	0.70 (H)	0.47 (M)
Kemerer	0.58 (H)	0.38 (M)	0.34 (M)	0.62 (H)	0.82 (H)	0.35 (M)	0.77 (H)
Kitchenham	0.29 (H)	0.43 (H)	0.45 (H)	0.48 (H)	0.34 (H)	0.35 (H)	1.08 (H)
Maxwell	0.59 (H)	0.63 (M)	0.41 (M)	0.59 (M)	0.57 (M)	0.46 (M)	0.38 (M)
Miyazaki	0.61 (H)	0.52 (H)	0.51 (H)	0.61 (H)	0.47 (H)	0.36 (H)	0.68 (H)
php_projects	0.62 (H)	0.47 (M)	0.43 (M)	0.62 (H)	0.78 (H)	0.48 (M)	0.99 (H)
telecom	0.68 (M)	0.70 (M)	0.37 (M)	0.68 (M)	0.68 (M)	0.70 (H)	0.29 (M)
USP05	0.86 (L)	0.27 (M)	0.27 (M)	0.86 (H)	0.24 (M)	1.49 (H)	1.60 (H)

Note: Effort values are recorded in the normalised form with their classification – High (H), Moderate (M) and Low (L).

Table 12
Comparison of the performance of the ATLM (state-of-the-art) and other regression models w.r.t. MAE.

	MAE (%)					
Dataset	**ATLM**	**RR**	**LASSO**	**ENR**	**RoR**	**SWR**
Albrecht	15.2	15.0	16.8	14.3	15.0	91.6
China	33.3	33.2	32.9	30.7	38.4	81.3
Cocomo81	25.5	17.8	11.1	36.6	11.2	43.4
Coconasav1	32.3	35.1	53.7	46.5	30.9	5.5
Coconasa2	50.2	49.6	77.1	63.6	48.0	32.4
Cosmic	30.9	37.1	34.7	31.5	16.5	70.3
Desharnais	41.6	43.4	67.8	36.3	42.4	72.0
Kemerer	33.7	16.8	15.4	41.3	35.6	35.4
Kitchenham	6.4	8.0	8.3	7.8	12.6	17.3
Maxwell	17.8	17.8	17.2	30.1	21.7	18.1
Miyazaki	10.5	12.0	12.9	12.9	21.4	22.7
php_projects	11.0	9.7	20.1	18.0	10.6	27.3
telecom	21.2	27.6	21.4	20.2	21.1	32.0
USP05	5.2	5.7	84.7	30.6	87.9	36.6
Sum of grey shaded cells	**3**	**1**	**3**	**4**	**1**	**2**

Note: The grey shaded cells indicate the best (minimum MAE) values across each dataset, recorded in percentages.

Table 13
Comparison of the performance of the ATLM (state-of-the-art) and other regression models w.r.t. BMMRE.

	BMMRE (%)					
Dataset	**ATLM**	**RR**	**LASSO**	**ENR**	**RoR**	**SWR**
Albrecht	11.8	24.5	21.6	25.8	27.0	82.4
China	65.0	57.2	86.7	45.2	53.3	53.8
Cocomo81	45.8	92.4	53.1	56.1	71.1	76.1
Coconasav1	37.9	68.0	74.0	54.7	63.0	61.7
Coconasa2	37.9	48.2	57.0	22.2	18.3	36.0
Cosmic	26.7	29.7	50.7	22.0	27.3	62.9
Desharnais	23.6	67.9	63.5	55.0	7.8	25.2
Kemerer	77.4	38.7	56.6	71.7	74.3	87.5
Kitchenham	57.4	71.0	89.5	46.0	76.2	49.8
Maxwell	54.2	64.9	81.0	27.1	54.9	61.2
Miyazaki	19.5	56.9	39.0	49.9	20.1	74.4
php_projects	48.8	45.9	71.9	67.7	52.9	36.7
telecom	66.7	59.5	54.3	8.1	78.2	79.0
USP05	49.0	78.0	66.7	36.3	40.1	49.9
Sum of grey shaded cells	**5**	**1**	**0**	**6**	**1**	**1**

Note: The grey shaded cells indicate best (minimum BMMRE) values across each dataset recorded in percentages.

assessment. We found that the ElasticNet regression (ENR), which is a hybrid regression-based model, performed better or slightly better than the ATLM [9] as measured by all three performance measures.

We agree with Whigham et al. [9] that a suitable regression- based model can form a baseline model for SEE. Therefore, we recommend the use of ENR as an auxiliary to the baseline ATLM [9] model. The results of our empirical experiment have shown that ENR, like ATLM, does not require the rigorous fine tuning of model parameters from one dataset to another. This makes it very simple and more robust for general purposes.

5. Conclusion

5.1. Summary of findings

This study introduces a classification scheme to aid in the self-guided interpretation of the software effort estimates that are predicted by estimation models. The density quantile function [26] is used to generate three effort classes (*high, low* and *moderate*) from 14 historical

Table 14
Comparison of the performance of the ATLM (state-of-the-art) and other regression models w.r.t. adjusted R^2.

	Adjusted R^2					
Dataset	**ATLM**	**RR**	**LASSO**	**ENR**	**RoR**	**SWR**
Albrecht	0.78	0.62	0.74	0.94	0.77	0.68
China	0.54	0.50	0.53	0.65	0.04	0.50
Cocomo81	0.56	0.78	0.45	0.69	0.46	0.77
Coconasav1	0.82	0.70	0.64	0.88	0.68	0.04
Coconasa2	0.33	0.28	0.18	0.75	0.12	0.46
Cosmic	0.38	0.79	0.31	0.80	0.02	0.68
Desharnais	0.54	0.33	0.49	0.49	0.24	0.03
Kemerer	0.83	0.16	0.69	0.71	0.74	0.74
Kitchenham	0.90	0.89	0.89	0.89	0.84	0.46
Maxwell	0.82	0.82	0.84	0.83	0.65	0.18
Miyazaki	0.89	0.88	0.88	0.87	0.73	0.79
php_projects	0.44	0.58	0.49	0.81	0.49	0.61
telecom	0.12	0.07	0.15	0.46	0.16	0.32
USP05	0.84	0.03	0.74	0.75	0.53	0.04
Sum of grey shaded cells	**5**	**1**	**1**	**7**	**0**	**0**

Note: The grey shaded cells indicate best (maximum adjusted R^2) values across each dataset. It should be noted that adjusted R^2 lies in the range [0–1].

datasets. Each of these software effort estimates are subsequently assigned to one of these classes to denote the effort expended during software development. The classification scheme is intended to minimise the problem of conclusion instability highlighted in previous studies [7,18,19,21]. The study addresses this challenge by estimating software effort (as *output 1*) and classifying the estimated effort into three levels or classes (as *output 2*). We call this estimation procedure the *duplex output software effort estimation.* The new classification scheme will allow researchers and software engineering practitioners to interpret software effort estimates in the context of the dataset used to develop the estimation models. Using three research questions, we first apply five feature subset selection techniques and three data transformation techniques (*box-cox,* log-*transform* and *z-score*) to 14 SEE datasets. The results show that the genetic search algorithm (GSA) and *z-score* normalisation have relatively significant effects on the prediction accuracy for all of the datasets. Thus, we recommend the use of the GSA for optimal feature subset selection for the *duplex output model.* Secondly, by incorporating the density quantile function, we are able to estimate the effort classes for all 14 SEE datasets. The results are validated with Welch's *t*-test at 95% and 99% confidence intervals. Lastly, with respect to the three performance measures, MAE, BMMRE and adjusted R^2, we observe that the ElasticNet regression has a better prediction accuracy than the baseline state-of-the-art model (ATLM) [9]. Thus, incorporating the GSA, *z-score* normalisation, density quantile function and ElasticNet regression into the *duplex output model* result in relatively high prediction accuracy. We conclude that, irrespective of the variations within SEE datasets, the *duplex output modelling* approach can minimise conclusion instability and improve prediction accuracy. Researchers and practitioners can adopt this approach to assess and easily interpret the software effort estimates generated by prediction models.

5.2. Threats to validity

5.2.1. External validity

We use 14 SEE datasets (13 of which are benchmark datasets in the SEE domain) that have been used in previous SEE studies [18,9,8,7,36]. These datasets are convenience samples, and are not generally representative of all SEE datasets. Therefore, the results from this study cannot be confidently generalised beyond these datasets. Again, the different characteristics, including data inconsistency, data diversity, heterogeneous data etc., of the 14 SEE datasets also pose challenges to

attempts to generalise the results. As these datasets emerge from multiple industries with different development policies, they may not be fair and generalisable representations of projects in a particular industry. Due to the heterogeneous nature of SEE datasets, it is important to note that some projects were eliminated during the preprocessing phase. For example, in the Desharnais dataset, four instances with missing entries were eliminated during preprocessing. Thus, the heterogeneous nature of SEE datasets, which leads to a reduction in the sampled projects, affects the generalisability of the results.

5.2.2. Construct validity

To evaluate whether the requisite measure(s) were taken to achieve construct validity, we consider six regression modelling approaches to build the SEE models: ATLM, stepwise regression, ridge regression, robust regression, ElasticNet regression and LASSO regression. The results indicate that most of the classifications are consistent across all six modelling techniques, supporting the validity of this research. As recommended by Kocaguneli and Menzies [18], this study adopts the leave-one-out (LOO) cross validation approach for training and validation purposes. As a result, other sampling methods, such as the 10-fold, 30-fold cross validation [9] and random splits [47] have not been assessed. However, we are confident that the LOO cross validation method is sufficient, as it has been effectively used in several previous effort estimation modelling studies. Three main performance measures, mean absolute error (MAE), balanced mean magnitude of relative error (BMMRE) and adjusted R^2 are used to assess the evaluations. Although other evaluation measures could have been used, these three measures have been proven to be reliable and effective in previous studies [24,10,23,32,71]. These performance measures are unbiased towards model overestimation and underestimation [23,71]. Other evaluation measures such as mean magnitude of relative error (MMRE), median magnitude of relative error (MdMRE) and PRED(k) are misleading evaluation measures that result in biases in model assessment [49]. It is difficult to establish the efficiency of data transformation techniques used in SEE studies, as so many have been used without actually assessing their relative performance. This study therefore evaluated three data transformation methods on the same set of datasets and found that the z-score transformation technique achieved the most accurate performance on all of the datasets used in this study.

5.2.3. Conclusion validity

The models used in this study are automated and this can result in automation bias. For example, the features are selected and the regression models are set up automatically. Automating a process involves making a series of assumptions that can result in biases. Nevertheless, the assumptions made in this study (e.g. data normality assumption) are based on the findings of previous studies [18,10,7] of building data mining and prediction models. Hence, the validity of the results can be trusted as the assumptions are reasonable. Finally, robust statistical tests (e.g. Welch's t-test and Kruskal-Wallis H-test) suitable for the datasets and problem at hand have been carefully applied to verify the validity of the results.

5.3. Future directions

Future research will examine projects from different repositories or organisations with the objective of increasing the generalisability of the results. We also plan to use other regression techniques such as the Bayesian regression to confidently benchmark our results. We also aim to expand our modelling approach by including non-regression-based models such as the deep learning approach and other statistical approaches to estimate the software effort.

Acknowledgement

This work is supported in part by the General Research Fund of the Research Grants Council of Hong KongNo. 11208017, and the research funds of City University of Hong Kong (No. 7004683 and 7004474).

References

[1] E. Mendes, N. Mosley, Bayesian network models for web effort prediction: a comparative study, IEEE Trans. Softw. Eng. 34 (6) (2008) 723–737.
[2] T. Menzies, Y. Yang, G. Mathew, B. Boehm, J. Hihn, Negative results for software effort estimation, Empirical Softw. Eng. (2016) 1–26.
[3] L.L. Minku, X. Yao, Software effort estimation as a multiobjective learning problem, ACM Trans. Softw. Eng. Methodol. 22 (4) (2013) 1–32.
[4] J. Huang, Y.-F. Li, M. Xie, An empirical analysis of data preprocessing for machine learning-based software cost estimation, Inf. Softw. Technol. 67 (June) (2015) 108–127.
[5] E. Kocaguneli, T. Menzies, J.W. Keung, On the value of ensemble effort estimation, IEEE Trans. Softw. Eng. 38 (6) (2012) 1403–1416.
[6] F. Sarro, A. Petrozziello, M. Harman, Multi-objective software effort estimation, 2016 IEEE/ACM 38th IEEE International Conference on Software Engineering (ICSE), 2016, pp. 619–630.
[7] J. Keung, E. Kocaguneli, T. Menzies, Finding conclusion stability for selecting the best effort predictor in software effort estimation, Autom. Softw. Eng. 20 (4) (2013) 543–567.
[8] E. Kocaguneli, T. Menzies, E. Mendes, Transfer learning in effort estimation, Empirical Softw. Eng. 20 (3) (2015) 813–843.
[9] P.A. Whigham, C.A. Owen, S.G. Macdonell, A baseline model for software effort estimation, ACM Trans. Softw. Eng. Methodol. 24 (3) (2015) 1–11.
[10] E. Kocaguneli, T. Menzies, J. Keung, D. Cok, R. Madachy, Active learning and effort estimation: finding the essential content of software effort estimation data, IEEE Trans. Softw. Eng. 39 (8) (2013) 1040–1053.
[11] C. Lokan, E. Mendes, Investigating the use of moving windows to improve software effort prediction: a replicated study, Inf. Softw. Technol. 56 (2014) 1063–1075.
[12] S. Amasaki, C. Lokan, A replication study on the effects of weighted moving windows for software effort estimation, Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering – EASE '16, 2016, pp. 1–9.
[13] E. Kocaguneli, T. Menzies, J.W. Keung, Kernel methods for software effort estimation: Effects of different kernel functions and bandwidths on estimation accuracy, Empirical Softw. Eng. 18 (1) (2013) 1–24.
[14] K. Dejaeger, W. Verbeke, D. Martens, B. Baesens, Data mining techniques for software effort estimation: A comparative study, IEEE Trans. Softw. Eng. 38 (2) (2012) 375–397.
[15] A. Corazza, S. Di Martino, F. Ferrucci, C. Gravino, F. Sarro, E. Mendes, Using tabu search to configure support vector regression for effort estimation, Empirical Softw. Eng. 18 (3) (2013) 506–546.
[16] D. Azhar, P. Riddle, E. Mendes, N. Mittas, L. Angelis, Using ensembles for web effort estimation, 2013 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), 2013, pp. 173–182.
[17] B. Kitchenham, E. Mendes, Why comparative effort prediction studies may be invalid, Proceedings of the 5th International Conference on Predictor Models in Software Engineering (PROMISE), ACM, 2009, pp. 1–5.
[18] E. Kocaguneli, T. Menzies, Software effort models should be assessed via leave-one-out validation, J. Syst. Softw. 86 (7) (2013) 1879–1890.
[19] I. Myrtveit, E. Stensrud, Validity and reliability of evaluation procedures in comparative studies of effort prediction models, Empirical Softw. Eng. 17 (1–2) (2012) 23–33.
[20] T. Menzies, M. Shepperd, Special issue on repeatable results in software engineering prediction, Empirical Softw. Eng. 17 (1–2) (2012) 1–17.
[21] B. Turhan, On the dataset shift problem in software engineering prediction models, Empirical Softw. Eng. 17 (1–2) (2012) 62–74.
[22] P. Phannachitta, J. Keung, A. Monden, K. Matsumoto, A stability assessment of solution adaptation techniques for analogy-based software effort estimation, Empirical Softw. Eng. 22 (1) (2016) 474–504.
[23] C. Lokan, E. Mendes, Investigating the use of moving windows to improve software effort prediction: a replicated study, Empirical Softw. Eng. 22 (2) (2016) 716–767.
[24] M. Shepperd, S. MacDonell, Evaluating prediction systems in software project estimation, Inf. Softw. Technol. 54 (8) (2012) 820–827.
[25] M.F. Bosu, S.G. Macdonell, Data quality in empirical software engineering: a targeted review, Proceedings of the 17th International Conference on Evaluation and Assessment in Software Engineering – EASE '13, 2013, pp. 171–176.
[26] R.L. Eubank, A Density-Quantile Function Approach to Optimal Spacing Selection, Ann. Stat. 9 (3) (1981) 494–500.
[27] C. Mair, M. Shepperd, M. Jørgensen, An analysis of data sets used to train and validate cost prediction systems, ACM SIGSOFT Softw. Eng. Notes 30 (4) (2005) 1.
[28] L. Song, L.L. Minku, X. Yao, The impact of parameter tuning on software effort estimation using learning machines, Proceedings of the 9th International Conference on Predictive Models in Software Engineering (PROMISE), ACM, 2013, pp. 1–10.
[29] S. Amasaki, C. Lokan, On the effectiveness of weighted moving windows: Experiment on linear regression based software effort estimation, J. Softw.: Evol. Process 27 (7) (2015) 488–507.
[30] B.W. Boehm, et al., Software Cost Estimation With Cocomo II, 1st ed., Prentice Hall PTR, Upper Saddle River, NJ, USA, 2000.
[31] J.W. Keung, B.A. Kitchenham, D.R. Jeffery, Analogy-X: providing statistical inference to analogy-based software cost estimation, IEEE Trans. Softw. Eng. 34 (4)

(2008) 471–484.
[32] C. Lokan, E. Mendes, Applying moving windows to software effort estimation, Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement (ESEM), IEEE Computer Society, 2009, pp. 111–122.
[33] B. Kitchenham, S.L. Pfleeger, B. McColl, S. Eagan, An empirical study of maintenance and development estimation accuracy, J. Syst. Softw. 64 (1) (2002) 57–77.
[34] S. Mensah, J. Keung, M. Bosu, K. Bennin, P.K. Kudjo, A stratification and sampling model for bellwether moving window, The 29th International Conference on Software Engineering and Knowledge Engineering (SEKE), 2017, pp. 481–486.
[35] S. Mensah, J. Keung, S.G. Macdonell, M.F. Bosu, K.E. Bennin, Investigating the significance of bellwether effect to improve software effort estimation, 2017 IEEE International Conference on Software Quality, Reliability and Security (QRS), 2017, pp. 340–351.
[36] S. Mensah, J. Keung, K.E. Bennin, M.F. Bosu, Multi-objective optimization for software testing effort estimation, The 28th International Conference on Software Engineering and Knowledge Engineering (SEKE), 2016, pp. 527–530.
[37] N.-H. Chiu, S.-J. Huang, The adjusted analogy-based software effort estimation based on similarity distances, J. Syst. Softw. 80 (4) (2007) 628–640.
[38] C. Mair, et al., An investigation of machine learning based prediction systems, The J. Syst. Softw. 53 (2000) 23–29.
[39] A. Heiat, Comparison of artificial neural network and regression models for estimating software development effort, Inf. Softw. Technol. 44 (15) (2002) 911–922.
[40] V. Khatibi Bardsiri, D.N.A. Jawawi, S.Z.M. Hashim, E. Khatibi, A flexible method to estimate the software development effort based on the classification of projects and localization of comparisons, Empirical Softw. Eng. 19 (4) (2014) 857–884.
[41] B. Turhan, E. Mendes, A comparison of cross-versus single-company effort prediction models for web projects, 2014 40th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), IEEE, 2014, pp. 285–292.
[42] T. Menzies, R. Krishna, D. Pryor, The Promise Repository of Empirical Software Engineering Data, North Carolina State University, Department of Computer Science, 2016 [Online]. Available: http://openscience.us/repo.
[43] B. Kitchenham, et al., Robust statistical methods for empirical software engineering, Empirical Softw. Eng. 21 (1) (2016) 212–259.
[44] E. Mendes, B. Kitchenham, Further comparison of cross-company and within-company effort estimation models for web applications, Proceedings of the 10th International Symposium on Software Metrics (METRICS'04), 2004, pp. 348–357.
[45] R.D. Cook, Detection of influential observation in linear regression, Technometrics 19 (1) (1977) 15.
[46] Y.S. Seo, D.H. Bae, On the value of outlier elimination on software effort estimation research, Empirical Softw. Eng. 18 (4) (2013) 659–698.
[47] J. Nam, S. Kim, Heterogeneous defect prediction, 10th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE), Bergamo, Italy, 2015, pp. 508–519.
[48] M.F. Bosu, S.G. Macdonell, A taxonomy of data quality challenges in empirical software engineering, Proceedings of the Australian Software Engineering Conference (ASWEC), IEEE, 2013, pp. 97–106.
[49] T. Foss, E. Stensrud, B. Kitchenham, I. Myrtveit, A simulation study of the model evaluation criterion MMRE, IEEE Trans. Softw. Eng. 29 (11) (2003) 985–995.
[50] Z. Chen, T. Menzies, D. Port, B. Boehm, Feature subset selection can improve software cost estimation accuracy, ACM SIGSOFT Software Engineering Notes, 30 2005, pp. 1–6.
[51] A. Arabali, M. Ghofrani, M. Etezadi-Amoli, M.S. Fadali, Y. Baghzouz, Genetic-algorithm-based optimization approach for energy management, IEEE Trans. Power Delivery 28 (1) (2013) 162–170.
[52] J. Andrews, T. Menzies, On the value of combining feature subset selection with genetic algorithms: faster learning of coverage models, Proceedings of the 5th International Conference on Predictor Models in Software Engineering (PROMISE), ACM, 2009, pp. 1–10.
[53] E.A. Hansen, S. Zilberstein, LAO*: a heuristic search algorithm that finds solutions with loops, Artif. Intell. 129 (1–2) (2001) 35–62.
[54] D.R. Morrison, E.C. Sewell, S.H. Jacobson, An application of the branch, bound, and remember algorithm to a new simple assembly line balancing dataset, Eur. J. Oper. Res. 236 (2) (2014) 403–409.
[55] M. Gütlein, E. Frank, M. Hall, A. Karwath, Large-scale attribute selection using wrappers, 2009 IEEE Symposium on Computational Intelligence and Data Mining (CIDM), IEEE, 2009, pp. 332–339.
[56] P. Bermejo, L. De La Ossa, J.A. Gámez, J.M. Puerta, Fast wrapper feature subset selection in high-dimensional datasets by means of filter re-ranking, Knowl.-Based Syst. 25 (1) (2012) 35–44.
[57] H. Liu, R. Setiono, A probabilistic approach to feature selection - a filter solution, Proc 13th International Conference on Machine Learning (ICML), 96 1996, pp. 319–327.
[58] P. Bertolazzi, G. Felici, P. Festa, G. Fiscon, E. Weitschek, Integer programming models for feature selection: new extensions and a randomized solution algorithm, Eur. J. Oper. Res. 250 (2) (2016) 389–399.
[59] F.H. Chen, H. Howard, An alternative model for the analysis of detecting electronic industries earnings management using stepwise regression, random forest, and decision tree, Soft Comput. 20 (5) (2016) 1945–1960.
[60] A.E. Hoerl, R.W. Kennard, Ridge regression: Biased estimation for nonorthogonal problems, Technometrics 12 (1) (1970) 55–67.
[61] R. Tibshirani, Regression shrinkage and selection via the lasso: a retrospective, J. R. Stat. Soc.: Ser. B (Stat. Methodol.) 73 (3) (2011) 273–282.
[62] H. Zou, T. Hastie, Regularization and variable selection via the elastic net, J. R. Stat. Soc.: Ser. B (Stat. Methodol.) 67 (2) (2005) 301–320.
[63] J. Friedman, T. Hastie, R. Tibshirani, Regularization paths for generalized linear models via coordinate descent, J. Stat. Softw. 33 (1) (2010) 1.
[64] P.T. Trish, O. Edison, P.H. Andrew, P. Giovanni, P. Carlo, Prediction of kinase inhibitor response using activity profiling, in vitro screening, and elastic net regression, BMC Syst. Biol. 8 (1) (2014) 74.
[65] Q. Li, B. Xie, J. You, W. Bian, D. Tao, Correlated logistic model with elastic net regularization for multilabel image classification, IEEE Trans. Image Process. 25 (8) (2016) 3801–3813.
[66] P.W. Holland, E.W. Roy, Robust regression using iteratively reweighted least squares, Commun. Stat.-Theory Methods 6 (9) (1977) 813–827.
[67] Y. Miyazaki, M. Terakado, K. Ozaki, H. Nozaki, Robust regression for developing software effort estimation models, J. Syst. Softw. 27 (1) (1994) 3–16.
[68] R.M. O'Brien, A caution regarding rules of thumb for variance inflation factors, Qual. Quan. 41 (5) (2007) 673–690.
[69] C.J. Willmott, K. Matsuura, Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance, Clim. Res. 30 (1) (2005) 79–82.
[70] A. Borji, M.-M. Cheng, H. Jiang, J. Li, Salient object detection: a benchmark, Image Process., IEEE Trans. 24 (12) (2015) 5706–5722.
[71] C. Kumar, D.K. Yadav, Software defects estimation using metrics of early phases of software development life cycle, Int. J. Syst. Assur. Eng. Manage., Springer India (2014) 1–9.
[72] T.W. MacFarland, J.M. Yates, Kruskal–Wallis H-test for oneway analysis of variance (ANOVA) by ranks, Introduction to Nonparametric Statistics for the Biological Sciences Using R, Springer International Publishing, 2016, pp. 177–211.
[73] B. Kitchenham, Keynote: robust statistical methods – why, what and how, Proceedings of the 19th International Conference on Evaluation and Assessment in Software Engineering (EASE), ACM, 2015, pp. 1–6.
[74] R. Krishna, T. Menzies, W. Fu, Too much automation? The bellwether effect and its implications for transfer learning, Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering (ASE), ACM, 2016, pp. 122–131.
[75] E. Kocaguneli, T. Menzies, How to find relevant data for effort estimation? 2011 International Symposium on Empirical Software Engineering and Measurement (ESEM), IEEE, 2011, pp. 255–264.