Software Qual J (2018) 26:1299–1325
https://doi.org/10.1007/s11219-017-9391-5

Application of mutual information-based sequential feature selection to ISBSG mixed data

Marta Fernández-Diego[1] ·
Fernando González-Ladrón-de-Guevara[1]

Published online: 27 October 2017

Abstract There is still little research work focused on feature selection (FS) techniques including both categorical and continuous features in Software Development Effort Estimation (SDEE) literature. This paper addresses the problem of selecting the most relevant features from ISBSG (International Software Benchmarking Standards Group) dataset to be used in SDEE. The aim is to show the usefulness of splitting the ranked list of features provided by a mutual information-based sequential FS approach in two, regarding categorical and continuous features. These lists are later recombined according to the accuracy of a case-based reasoning model. Thus, four FS algorithms are compared using a complete dataset with 621 projects and 12 features from ISBSG. On the one hand, two algorithms just consider the relevance, while the remaining two follow the criterion of maximizing relevance and also minimizing redundancy between any independent feature and the already selected features. On the other hand, the algorithms that do not discriminate between continuous and categorical features consider just one list, whereas those that differentiate them use two lists that are later combined. As a result, the algorithms that use two lists present better performance than those algorithms that use one list. Thus, it is meaningful to consider two different lists of features so that the categorical features may be selected more frequently. We also suggest promoting the usage of Application Group, Project Elapsed Time, and First Data Base System features with preference over the more frequently used Development Type, Language Type, and Development Platform.

Keywords Feature selection · Mutual information · ISBSG · Software development effort estimation · *k*-nearest neighbor

✉ Marta Fernández-Diego
marferdi@omp.upv.es

Fernando González-Ladrón-de-Guevara
fgonzal@omp.upv.es

[1] Department of Business Organisation, Universitat Politècnica de València, 46022 Valencia, Spain

1 Introduction

Generally, software development effort estimation (SDEE) requires data in a mathematically feasible format through Data Pre-processing (DPP). DPP techniques consist of data reduction, data projection, and missing data treatment. Data reduction aims to decrease the size of the datasets by means of feature selection (FS) or case selection (Huang et al. 2015). FS reduces the effects of high dimensionality on a dataset by selecting an optimum subset of features and removing the rest of the features from further analysis (Awada et al. 2012). The removed features are either irrelevant to the problem at hand or redundant when compared to the features within the optimum subset. Besides, the computation cost of subsequent analysis is reduced.

Anyway, there is little research work focused on DPP and specially FS techniques in the SDEE literature (Huang et al. 2015). The first attempt of applying FS to parametric SDEE models (Chen et al. 2005) showed that it can dramatically improve cost estimation. In (Liu et al. 2013), the Desharnais dataset is used for experiments but other datasets such as ISBSG should be considered too.

Existing FS approaches are mainly designed for classification problems with categorical or continuous features (Liu and Yu 2005). However, in SDEE, the data collected include both categorical and continuous features. One approach to deal with mixed data is to perform a discretization for continuous features (Hall and Holmes 2003; Ferreira and Figueiredo 2011).

Hence, this paper aims to address the problem of selecting the most relevant features from ISBSG (International Software Benchmarking Standards Group) dataset to be used in SDEE. To deal with mixed features, this paper follows the approach presented by Doquire and Verleysen (2011) to obtain a ranked list of features. This list is then split into two regarding the continuous and categorical features and next recombined according to the accuracy of a case-based reasoning (CBR) model.

In the first part of this paper, the theoretical background is presented along with the description of the four proposed FS algorithms. Next, the methodology is considered, followed by the results obtained when applying the different FS algorithms regarding their performance and the resulted selected features. Finally, the paper concludes with a brief discussion on the threats to validity, conclusions, and future lines of research.

2 Mutual information-based feature selection methods

2.1 Background

FS is the focus of research in areas of application such as clustering or classification and is used in various domains including software effort estimation (Song and Shepperd 2007). The objective of feature selection is threefold: improving the predictors' performance, providing faster and more cost-effective predictors, and facilitating a better understanding of the underlying process that generated the data (Guyon and Elisseeff 2003).

FS aims to choose a subset of the original features according to a selection criterion, so that the redundancy and noise in the original feature set is minimized (Liu et al. 2014). On the one hand, by eliminating irrelevant features, the remaining ones are useful for estimation purposes (Dejaeger et al. 2012). On the other hand, redundant features are those features that depend on other features (Liu et al. 2013). Hence, it is advised to focus on data quality rather than

collecting as many predictive features as possible. The aim is to build an estimation model that performs better than that built with the original features.

The search strategy and the evaluation criteria work together to find the best feature subset (Gupta et al. 2014). According to the search strategy, global (Somol et al. 2004), heuristic (Dash and Liu 2003), and random (Oh et al. 2004) strategies were introduced in the literature (Kabir et al. 2011). In this regard, the search engine can be divided into three categories: exhaustive, heuristic, and non-deterministic search engines (Guyon and Elisseeff 2003). With respect to the evaluation criteria, FS approaches can be classified into three categories (Kabir et al. 2011): the filter, the wrapper, and the hybrid approach.

FS methods that make use of a proxy measure to estimate utility are termed "filter" approaches. Filter-based FS methods rely on various measures of the training data, such as distance, information, dependency, and consistency to measure the correlation between features (Liu and Motoda 2012). FS methods that assess feature utility with respect to a given classifier or clustering method are referred to as "wrapper" approaches (Kohavi and John 1997; Mandal and Mukhopadhyay 2013). Wrapper methods differ from filter-based FS in that they use a learner when evaluating the features, either separately or as subsets.

Filter methods have good generalization properties and are computationally cheaper than the wrapper approaches but may be less effective at decreasing dimensionality. Usually, wrappers can yield high fitting accuracy at the cost of high computational complexity (especially for subset evaluation) and low generalization of the selected features to other conditions (Li et al. 2009). In the hybrid approach, features are first filtered and then determined by the wrapper model (Hsu et al. 2011). It is often found that the hybrid approach is capable of locating a good solution.

In filter methods, information theory (Shannon 1949) has been widely applied. Measures such as mutual information (MI), interaction information, conditional mutual information, and joint mutual information can be used to compute the relevance of features (Bennasar et al. 2015). MI is a criterion from the information theory which has proven to be very efficient in feature selection (Battiti 1994; Fleuret 2004) mainly because it is able to detect non-linear relationships between variables. MI can be expressed as the amount of information provided by variable X which reduces the uncertainty of variable Y, i.e., the amount of information both variables share. It is formally defined as follows:

$$I(X;Y) = H(X) + H(Y) - H(X,Y) \tag{1}$$

where $H(X)$ is the entropy with Shannon definition given by

$$H(X) = -\int f_X(x) \log f_X(x) dx \tag{2}$$

with f_X being the probability density function of X. $H(X, Y)$ is the entropy of the joint variable (X, Y) defined in the same way.

$$\text{MI is symmetric, hence : } \quad I(X;Y) = I(Y;X) \tag{3}$$

$I(X; Y) = 0$ if and only if X and Y are independent random variables.

$$I(X;Y) \text{ is bounded by the individual entropies of } X \text{ and } Y : I(X;Y) \leq \min\{H(X), H(Y)\} \tag{4}$$

MI can be reformulated as

$$I(X;Y) = \iint f_{X,Y}(x,y) \log \frac{f_{X,Y}(x,y)}{f_X(x) f_Y(y)} dxdy \tag{5}$$

Due to the fact that in practice none of the probability density functions f_X, f_Y, and $f_{X,Y}$ are known, MI cannot be computed analytically but has to be estimated from the data set. Namely, when at least one of the variables X and Y is continuous, their mutual information $I(X;Y)$ is hard to compute, because it is often difficult to compute the integral in the continuous space based on a limited number of samples. One solution is to incorporate data discretization as a pre-processing step (Peng et al. 2005).

MI is used as a measure of both the relevance of a feature which is to be maximized and the feature redundancy which is to be minimized (Mandal and Mukhopadhyay 2013). It is mostly used because it does not require to assume knowing the sample distribution, does not need to transform the data, and can measure the degree of uncertainty between features in a quantified form.

For example, Battiti (1994) proposed a greedy selection method called MIFS (Mutual Information Feature Selection) that selects the feature that maximizes the information about the class, corrected by subtracting a quantity proportional to the average MI. Kwak and Choi (2002) proposed a method called MIFS-U (Mutual Information Feature Selector Under Uniform Information Distribution) that makes a better estimation of MI between input features and output classes. Nonetheless, a user-defined parameter is used in MIFS and MIFS-U, and it may cause errors in results.

To avoid this kind of error, mRMR (Minimum Redundancy Maximum Relevance) method (Peng et al. 2005) is proposed. In mRMR, the user-defined parameter is replaced by the reciprocal of selected feature numbers. However, in the presence of many irrelevant and redundant features, mRMR has limitations. NIMFS (Normalized Mutual Information Feature Selection) method (Estévez et al. 2009) is then proposed by defining the normalized MI as a measure of redundancy between one feature and the selected subset. INMIFS (Improved NMIFS) method (Vinh et al. 2010) also restricts the value of relevance part into [0,1] to balance the relevance and redundancy components.

The problem of the MI-based correlation measure is that it can only be defined between continuous or categorical features. Thus, these FS methods are designed to work only with continuous or categorical features. Actual data, however, usually include mixed features that include both categorical and continuous features, for example, in SDEE. A simple approach to handle the problem of mixed features is to turn it into a discrete or continuous one, but both approaches have drawbacks.

In their method, Kwak and Choi (2002) already dealt with between mixed features, with the assumption that all samples had the same probability of occurrence. They divided the continuous input feature space into several discrete partitions and calculated the MI using the definitions for discrete cases. In (Liu et al. 2013), a new method (C-mRMR) for calculating MI between mixed features is proposed with the latter assumption removed. Since MI tends to choose features with more values, by normalizing this correlation measure, the mRMR criteria is improved to deal with mixed data. In order to eliminate the different influences, all types of features (numerical, ordinal, and nominal) are normalized in (Li et al. 2009) where the MICBR (Mutual Information Feature Selector for Case Based Reasoning) algorithm proposed consists of an inner stage at which the classical MIFS (Battiti 1994) is used to rank the features and an outer stage which fixes the number of features by minimizing the error. Finally, the approach proposed by Doquire and Verleysen (2011) to deal with mixed data is different: the features of each type are first ranked independently and two lists are produced. These lists are then combined according to the accuracy of a classifier.

2.2 Proposed feature selection algorithms

The proposed FS algorithms use MI as a measure of both the relevance of a feature and its redundancy. The function *information.gain* from FSelector package (Romanski and Kotthoff 2014) has been employed in this paper to obtain a measure of MI. Its syntax is *information.gain(formula*, *data)* where *formula* is a symbolic description of a model and *data* represents the data to be processed. In fact, this function is calculated as $H(Class) + H(Attribute) - H(Class, Attribute)$. The FSelector package in turn imports the package Entropy (Hausser and Strimmer 2009) that implements various estimators of the Shannon entropy. The function *information.gain* internally discretizes the continuous features, as a previous step, by means of the Fayyad and Irani's Minimum Description Length method (Fayyad and Irani 1993) that is one of the most commonly used discretization methods in machine learning (Witten et al. 2011). This univariate supervised discretization method combines an entropy-based splitting criterion with a minimum description length stopping criterion to determine the best cutpoint for splitting an interval (Lustgarten et al. 2008).

The major difference between the proposed FS algorithms and the approach followed in (Doquire and Verleysen 2011) consists on the application of this common MI measure to construct the ranking of both continuous and categorical features, provided that the continuous features are first discretized. As a precedent, the mrMR method has been proved to attain adequate results with a previous step of feature discretization (Ferreira and Figueiredo 2011).

The FS algorithms considered in this work are depicted in Fig. 1 and described below:

- MI_1L. This algorithm employs a forward search strategy that is a quite simple and fast solution to search the feature space and build the set of selected features. MI_1L consists in adding, at each step, the best possible feature according to a specific criterion: The features are ranked according to the relevance with respect to the dependent feature using MI as a measure. This ranking is obtained once on the whole dataset. Then, all features are tested sequentially considering the aforementioned ranked list (1L). The inclusion of the features in the set of selected features is determined by a predictive model: if the inclusion of a feature improves the accuracy of the previous CBR model in terms of MMRE, this feature will be added to the list of selected features; otherwise, it will be excluded. Therefore, this algorithm combines aspects of a filter method that provides a feature ranking and is used as a pre-processing step, and a wrapper method that makes use of a predictive model to assess feature subsets.
- mRMR_1L. Nevertheless, it could be argued that a feature very relevant to the dependent feature may be useless when it conveys similar information to that provided by another selected feature. Thus, it should not be selected. To solve this, at each step, we can search for the feature that maximizes the difference between its relevance and its redundancy with the already selected features. Indeed, the only difference with MI_1L algorithm is that the features are initially ranked according to mRMR criterion (Minimum Redundancy Maximum Relevance (Peng et al. 2005)). For feature $Xi (i \notin S)$ given a dependent feature Y, where S is the set of indices of already selected features, mRMR can be expressed as

$$mRMR(Xi) = I(Xi; Y) - \frac{1}{|S|} \sum_{j \in S} I(Xi; Xj) \tag{6}$$

where $I(Xi; Y) = D(Xi)$ is the estimated relevance and $\frac{1}{|S|} \sum_{j \in S} I(Xi; Xj) = R(Xi)$ is the estimated redundancy.

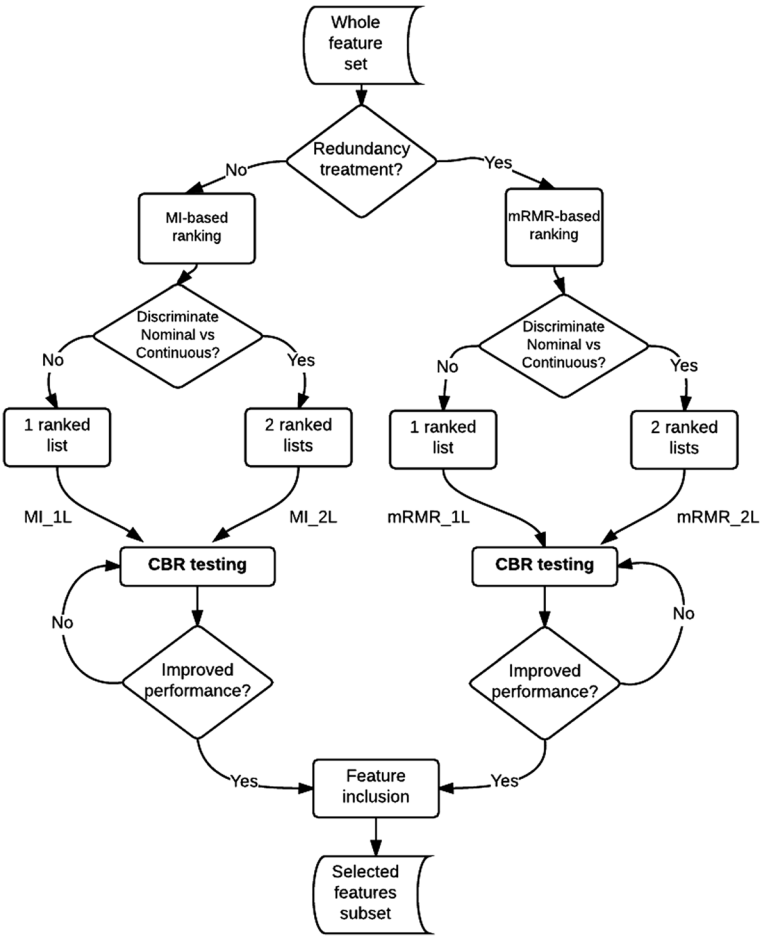

Fig. 1 Block diagram of proposed FS algorithms

In other words, if a subset of features has already been selected, the next unselected feature *Xi* to be tested is the one which maximizes mRMR. The relevance *D* is obtained as the MI between the new feature *Xi* and the dependent feature *Y*, while the redundancy *R* can be calculated as the average MI between the new feature and each of the already selected features ($Xj \in S$). This ranking will be used later by the greedy forward search strategy.

- MI_2L. Due to the fact that the model involves categorical and continuous features, it seems interesting to make a distinction between them. Consequently, the features of each type are ranked separately in two independent lists where only the relevance, based on the same MI measure, is taking into account. These lists are then combined according to the accuracy of a CBR model. Thus, both the top-ranked categorical and continuous features are considered at each step. The one whose addition to the current set of features leads to the best prediction accuracy of the CBR model is chosen and removed from its list. Anyway, the discarded feature still has the possibility to compete with the next best feature of the other list. This wrapper procedure ends when all features have been tested.

- mRMR_2L. This algorithm uses the mRMR criterion and also discriminates between categorical and continuous variables.

The block diagram of Fig. 1 serves to contrast the aforementioned algorithms.

On the one hand, the algorithms MI_1L and MI_2L just consider the relevance (MI) of any independent feature with respect to the dependent feature (NWEL1), while mRMR_1L and mRMR_2L follow the mRMR criterion of maximizing relevance and also minimizing redundancy between any independent feature and the already selected features. On the other hand, the algorithms that do not discriminate between continuous and categorical features consider just one list (MI_1L and mRMR_1L), whereas those that discriminate between them use two lists (MI_2L and mRMR_2L) following (Doquire and Verleysen 2011).

3 Methodology

Firstly, the pre-processing procedure on ISBSG Release 12 is described in this section, leading to a data subset that includes 621 projects and 12 features. Then, the four proposed FS algorithms, introduced in Section 2.2, are compared to select the most relevant and non-redundant mixed features for SDEE. MI_1L and MI_2L algorithms just maximize relevance, while mRMR_1L and mRMR_2L algorithms minimize redundancy too. MI_2L and mRMR_2L use two lists regarding continuous and categorical features. Additionally, a greedy forward feature selection algorithm is introduced as a baseline.

All statistical computations and graphics were created with the open source software package R (R Core Team 2015) using version 3.2.3 with additional packages (FSelector, gdata, ggplot2, dplyr, tidyr, and VIM).

3.1 Data Pre-processing procedure

3.1.1 ISBSG

The International Software Benchmarking Standards Group (ISBSG 2013a) designed and maintains two international public repositories (Software Development & Enhancement and Maintenance & Support) to improve management of IT resources by both business and government. The ISBSG dataset offers a wealth of information about completed software projects, regarding practices, tools, and methodologies, accompanied by process and product data, to be used in benchmarking, monitoring, quality control, and performance management purposes during the software development process (Top et al. 2011). However, there are some issues that need to be considered when using it (Fernández-Diego and González-Ladrón-de-Guevara 2014). The experimental work in this paper is based on ISBSG Release 12 which includes 6006 projects and 126 features.

3.1.2 Filtering

Given that ISBSG is a large and heterogeneous dataset, a data preparation process is required before applying any analysis. The filtering rules in Table 1 were adapted from (Lokan and Mendes 2009a).

Table 1 Selection criteria for effort concerns

Selection criteria	Projects remaining	Projects removed
High data quality		
High general data quality[a]	5558	448
High functional size quality[b]	3935	1623
Comparable effort definition		
Development team effort known[c]	3215	720
Effort across the whole life cycle[d]	2249	966
Comparable size definition		
IFPUG version 4.0 or later[e]	1884	365

[a] ISBSG <- ISBSG[ISBSG$Data.Quality.Rating %in% c("A","B"),]

[b] ISBSG <- ISBSG[ISBSG$UFP.rating %in% c("A","B"),]

[c] ISBSG <- ISBSG[!is.na(ISBSG[["Normalised.Work.Effort.Level.1"]]),]

[d] ISBSG <- ISBSG[ISBSG$Normalised.Work.Effort.Level.1 == ISBSG$Summary.Work.Effort,]

[e] ISBSG <- ISBSG[ISBSG$Count.Approach=="IFPUG 4+",]

3.1.3 Initial set of features

Three effort features are recorded in the ISBSG dataset. The fundamental feature is Summary Work Effort (SWE), measured in staff hours. It is the total effort for the project, as reported by the contributing organization, but SWE could not cover all life cycle phases of the project. The other two effort features take into account differences in whose effort is included in SWE, and which life cycle phases are included in SWE: Normalised Effort is ISBSG's estimate of the total effort when any "missing" phases are added. However, there can still be some inconsistency between projects, even when using Normalised Effort, because projects report effort involving different participants and this is indicated by the variable Resource Level. Level 1 means that effort is reported for the development team only, levels 2 and 3 add effort for development team support and computer operations involvement, and level 4 adds effort for end users and clients. Consequently, Normalised Work Effort Level 1 (NWEL1) is the normalized effort for the development team only. Hence, to ensure maximum consistency, González-Ladrón-de-Guevara et al. (2016) recommend the use of NWEL1 as dependent variable, which is the one selected in this paper.

For a start, this study focuses on the 20 ISBSG features most frequently used as independent variables in effort estimation models according to (González-Ladrón-de-Guevara et al. 2016). However, since the experimental work is based on ISBSG R12, two features, Application Type (AT) and Organization Type (OT) have been replaced by their respective derived features Application Group (AG) and Industry Sector (IS). AT identifies the type of application within the business area and organization/industry type being addressed by the project, while OT identifies the type of organization that submitted the project. When using previous ISBSG releases, the values of these variables are usually regrouped. This categorization, however, could be complex. Thus, the new features in R12, AG, and IS intend to group the values of AT and OT into a predefined set of values to overcome the aforementioned handicap for both researchers and practitioners.

Secondly, from this initial set of the 20 most used features, those with missing values larger than 60% have been excluded from further analysis: Average Team Size (87.31%), Business Area Type (70.49%), Max Team Size (62.47%), and Input count, Output count, Enquiry count, File count, and Interface count (61.68%). During the pre-processing stage, it is a common

practice to prune features due to a high level of missing data (Huang et al. 2008; Jeffery et al. 2001; Lokan 2005; Mendes et al. 2005) when no imputation treatment is performed.

Thirdly, the selection criteria in Table 1 that provides a comparable definition for effort ensures, on the one hand, that NWEL1 has no missing values and on the other hand, that all values of feature Resource Level are equal to one. For this reason, Resource Level also dropped at this stage. Indeed, this variable is more often used in the filtering process than in the estimation model itself (González-Ladrón-de-Guevara et al. 2016). Hence, the subset includes 1884 projects with 11 independent variables and the dependent one (NWEL1).

Finally, projects with any missing value in the independent variables were discarded. After the pre-processing stage, the resulting dataset includes 621 projects and 12 features that are shown in Table 2. The independent features are as follows:

- Adjusted Function Points (AFP) is the adjusted size for IFPUG, NESMA, FiSMA, and MARK II counts. The functional size is adjusted by a Value Adjustment Factor and the resultant adjusted size is reported as AFP. It has a mean value of 461.7 and a standard deviation of 902.8 adjusted function points.
- Application Group (AG) is a derived field that groups Application Type of the project into a single value of a defined set: Business Application (94.85%), Real-Time Application (3.06%), Mathematically Intensive Application (1.29%), and Infrastructure Software (0.80%).
- 1st Data Base System (1DBS) is the primary database used in a project. This feature will be considered later when dealing with categorization.
- Development Platform (DP) defines the primary development platform, determined by the operating system used. Each project is classified as PC (16.75%), Mid Range (9.18%), Mainframe (34.30%), or Multi-platform (39.77%). This variable is clearly determined at the early stage of any software project. DP is the best indicator of the environment in which a project is developed and does not refer specifically to the hardware platform (Hill 2010).
- Development Type (DT) describes whether the development was a New Development (41.22%), Enhancement (56.68%), or a Re-development (2.1%). A re-development is similar to a new development, using new technologies to replace or upgrade an existing software product. This variable has no missing values. It is one of the most important criteria for selecting projects (Lokan and Mendes 2012) and is suggested by ISBSG guidelines for use in the estimation process as well as for benchmarking (ISBSG 2013b). Empirical evidence exists that the development type influences project effort (Huang et al. 2008; Moses et al. 2006).
- Functional Size (FSZ) represented the size in AFP up to and including Release 8. Since Release 9 (released in 2004), it represents the unadjusted function point count (UFP), which reflects the specific countable functionality provided to the user by the project or application (ISBSG 2013c) before any adjustment. FSZ and AFP have been reported separately in the dataset since Release 9. FSZ has a mean of 449.8, and a standard deviation of 785.8 function points.
- Industry Sector (IS) identifies the type of organization that submitted the project. The possible of values of IS are as follows: Banking (10.63%), Communication (10.14%), Construction (1.29%), Electronics & Computers (1.77%), Financial (4.35%), Government (28.66%), Insurance (16.59%), Manufacturing (9.02%), Medical & Health Care (0.97%), Mining (0.81%), Professional Services (2.42%), Service Industry (9.50%), Utilities (2.25%), Wholesale & Retail (1.61%).

Table 2 Selected ISBSG features

Feature	Identifier	Type	States
Adjusted Function Points	AFP	Continuous	[6..17518] AFP
Application Group	AG	Categorical	Business Application, Infrastructure Software, Mathematically-Intensive Application, Real-Time Application
1st Data Base System	1DBS	Categorical	ACCESS, ADABAS, Attain, DB2, Domino, Exchange, Foxpro, HIRDB, IMS, MS SQL, NCR, No, ORACLE, SAS, Solid, SYBASE, Unspecified, Watcom, WGRES
Development Platform	DP	Categorical	MF, MR, Multi, PC
Development Type	DT	Categorical	Enhancement, New Development, Re-development
Functional Size	FSZ	Continuous	[6..13580] FP
Industry Sector	IS	Categorical	Banking, Communication, Construction, Electronics & Computers, Financial, Government, Insurance, Manufacturing, Medical & Health Care, Mining, Professional Services, Service Industry, Utilities, Wholesale & Retail
Language Type	LT	Categorical	3GL, 4GL, ApG
Normalised Work Effort Level 1	NWEL1	Continuous	[26..71729] hours
Primary Programming Language	PPL	Categorical	.Net, AB INITIO, ABAP, Access, ASP, C, C/AL, C#, C++, COBOL, Datastage, DELPHI, EASYTRIEVE, HTML, Java, JavaScript, Lotus Notes, NATURAL, Oracle, OutlookVBA, PL/I, PL/SQL, PowerBuilder, Pro*C, RPG, SAS, Shell, SQL, TELON, Unspecified, Visual Basic, Visual Studio .Net
Project Elapsed Time	PET	Continuous	[0.4..87] months
Used Methodology	UM	Categorical	Don't know, No, Yes

- Language Type (LT) defines the programming language type used for the project. In the subset, third-generation languages dominate (67.79%) and fourth-generation languages are also well represented (31.56%), but Application Generator (0.65%) is hardly represented. Statistical evidence exists indicating that LT has an impact on effort (Lokan and Mendes 2009b). In practice, high-level programming languages and in particular all 4GL languages are designed to reduce programming effort, but in contrast, they require considerable effort during the design phase (Chatzipetrou et al. 2012).
- Project Elapsed Time (PET) represents the total elapsed time for the project in calendar months (actual duration). This variable is related to the effort on a software project, when considered with the resources that have been allocated. These resources are to some extent reflected by the team size and dedication of the team. PET has a mean value of 8.04 months, a median value of 6 months, and a standard deviation of 6.9 months. Moreover, it has been used too as an independent variable, similarly to other variables regarding the duration of the project such as Project Inactive Time, or variables concerning the team size (Maximum Team Size and Average Team Size).
- Primary Programming Language (PPL) indicates the primary language used for development. Since particular programming languages belong to one of the language types, this variable is in a way redundant with LT (Jiang and Comstock 2007). When two or more independent variables contain redundant information, only one is to be considered (Bibi et al. 2008; Huang and Chiu 2006). LT is more often used than PPL, except where information about the specific programming language is required. The categorization of this variable will be considered in the next section.
- Used Methodology (UM) states whether (56.04%) or not (4.51%) a development methodology was used by the development team to build the software ("Don't know" accounts for the remaining 39.45%).

3.1.4 Categorization

Some of these features have too many distinct levels or are not recorded in a consistent format. Hence, there is a need to formalize these features so as to minimize confusion of concepts and to maximize both consistency and, when possible, the number of responses for each level (Deng and MacDonell 2008). The formalization rules are described in the next paragraphs and the resulting levels of the categorical features are recorded in the fourth column of Table 2.

Particularly, two features have been re-categorized: PPL and 1DBS. PPL presented 36 levels after filtering. In this regard, two projects out of 621 with unclear values have been transmuted to "Unspecified" (Deng and MacDonell 2008). Besides, PPL values corresponding to other three projects were converted into more general programming languages (ASP, C++, and Visual Basic). These minor changes resulted in obtaining 32 levels. These values are coherent with those presented in (ISBSG 2013c). The most relevant programming languages are as follows: COBOL (19.48%), Visual Basic (18.36%), PL/I (12.56%), and Java (8.37%).

If known, 1DBS is the primary technology database used to build or enhance the software (i.e., that used for most of the build or enhancement effort); nevertheless, there are 21 projects (3.38%) that do not use a DBS. This feature is not normalized and includes simply descriptive strings rather than predefined categories. Hence, the number of distinct levels, after filtering, is 83. Besides, 1DBS values are not recorded in a consistent format. To start with, some values

such as "Yes," "multiple," "ISAM," and "VSAM" were transformed into "Unspecified" (6.44%). Next, DBS categories were formed by grouping together related occurrences (i.e., Oracle 7, Oracle 7.3, Oracle 7.3.4, Oracle 8, Oracle 8.0, and so on, formed Oracle category; DB2, DB2/2, DB2 V3, DB2 UDB, and similar ones, formed DB2 category, etc.). As a result, 19 categories have been obtained. The most relevant DBS are as follows: DB2 (34.46%), ORACLE (26.25%), IMS (12.24%), MS SQL (7.41%), and Access (3.54%).

3.2 Case-based reasoning in software development effort estimation

The case-based reasoning (CBR) approach was firstly proposed by Shepperd and Schofield (1997) as a valid alternative to expert judgment and algorithmic method for SDEE. CBR works similarly to the way in which an expert typically estimates software effort (Dejaeger et al. 2012), i.e., the most similar historical projects are selected to estimate a new project using a similarity measure. Furthermore, it is a convenient methodology when dealing with mixed features (Liu et al. 2013). Generally, there are three parameters for CBR method: the similarity measure, the number of most similar projects (analogies), and the analogy adaptation (Angelis and Stamelos 2000).

As one of the key components of CBR, the similarity measure considers the level of similarity between projects. Several similarity measures have been proposed, such as the Euclidean, Manhattan, Clark, or Canberra distance. The performance of these similarity measures is strongly related to the type of features representing each project. It is quite different to deal with continuous or categorical data (Núñez et al. 2004). These measures are also sensitive to irrelevant, interacting, or noisy features. Hence, the choice of project features has large impact on the similarity measure and FS has become an important pre-processing step that generally improves CBR, especially for SDEE (Mendes et al. 2003) when using ISBSG, Desharnais, and Kitchenham datasets (Huang et al. 2015). Several FS methods have been proposed for CBR, such as exhaustive search (Shepperd and Schofield 1997), hill climbing, and forward sequential selection (Kirsopp et al. 2002). However, most existing feature selectors for CBR are the so-called wrappers (Kohavi and John 1997).

The number of analogies considers the number of most similar projects that will be used to generate the estimation. In particular, one or more historical software projects are needed to estimate the cost of a new project. *k*-nearest neighbor (*k*-NN) is a classification technique that should be one of the first choices for a classification study when there is little prior knowledge about the distribution of the data (Dudani 1976). This technique is broadly used in CBR. *k*-NN is commonly based on the Euclidean distance between a test sample and the specified training samples. *k*-NN is also quite sensitive to the considered features, i.e., it is less effective when many features are irrelevant or noisy. Many studies suggest the closest analogy ($k = 1$) but, in this study, we have considered $k = \{1, 2, 3, 4\}$ to cover the most commonly used values (Huang and Chiu 2006; Jørgensen et al. 2003; Mendes et al. 2003; Shepperd and Schofield 1997).

After the analogies have been selected, the prediction for a new project is determined by certain statistics based on the *k* selected projects. The closest analogy, the mean of closest analogies, the median, and the inverse distance weighted mean (Kadoda et al. 2000) are the most common adaptation techniques. In this paper, however, the mean of the *k*-nearest neighbors is used to estimate software project costs. Using this measure of central tendency, all analogies are treated as being equally influential on the cost estimates (Li et al. 2009).

3.3 Comparison of Feature Selection algorithms

The four FS algorithms described in Section 2.2 were compared and also tested against a baseline. In this regard, a greedy forward selection (GFS from here on) algorithm was used to compare the accuracy and computational efficiency of the proposed forward selection algorithms based on one or two ordered lists.

The GFS procedure begins by evaluating all *M* feature subsets which include just one feature to find the best one. Next, forward selection finds the best two-feature subset, including the variable selected in the first loop, and a second one from the remaining features. Hence, there are *M - 1* pairs to be tested. The new feature contained in this pair is included only when the best two-feature CBR outperforms the best CBR obtained from the first loop. This procedure consequently continues assessing new features accordingly to the aforementioned inclusion criteria.

For comparison purposes, the prediction accuracy and the computational cost of the FS algorithms were considered. Furthermore, both the number of selected features and the preference of usage of these features were examined.

3.4 Multiple 3-fold cross-validation

Cross-validation is usually used when testing for classification accuracy (Awada et al. 2012) by splitting the data into multiple *n* folds or partitions of equal size. The first $n-1$ folds are used for training the learner, and the remaining fold is used to test the learner. This process is performed *n* times so that each of the folds will be used as the test fold. In this study, the number of splits to make in the dataset was set to three. Hence, two folds are used to construct the estimation model and the remaining fold tests the performance of the model using the Mean Magnitude of Relative Error (MMRE). Hall and Holmes (2003) recommend repeating the *n*-way cross-validation multiple times, each time randomizing the order of the data to reduce order effects (influenced by the ordering of the data) and perhaps also excessive variance. Cross-validation is also run multiple times to compare the performance of the FS algorithms using statistical significance tests such as the Wilcoxon rank test. Hence, in this paper, cross-validation was run 500 times regarding the convergence of the algorithms. To sum up, for each run, the whole data set (621 projects) is considered but different seeds are used to obtain random 3-fold splits.

4 Experimental results

This section presents the experimental results. Basically, the performance of the four FS algorithms is compared between them and against the GFS algorithm, first considering accuracy issues, and finally, by analyzing the selected features.

4.1 Accuracy of feature selection algorithms

4.1.1 Convergence of the algorithms

First of all, the convergence of the algorithms was analyzed. To quantify the variation, cross-validation was repeated 500 times so as to estimate the distribution of the performance statistics. This allowed us to fix the number of cross-validations to be considered in the experimental work.

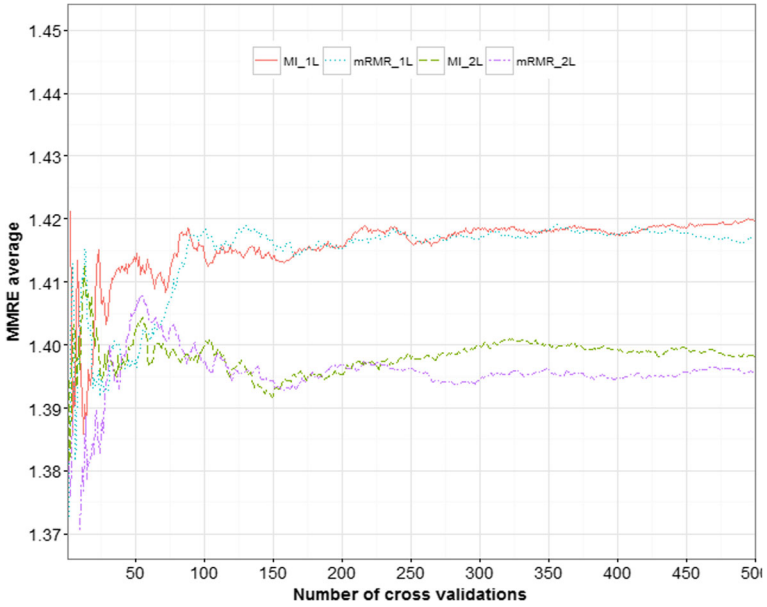

Fig. 2 Evolution of the cumulative means of MMRE values for $k = 1$

Figure 2 shows the evolution of the MMRE average as a function of the number of cross-validations performed for each algorithm and $k = 1$. Similar figures and analysis were obtained for other values of k. The results denote that FS algorithms are highly sensitive to data. This can be seen in Fig. 2 where MMRE cumulative means fluctuate when considering the first 100 cross-validations and progressively consolidate. In fact, data series present minor fluctuations after 300 runs.

To support the algorithm convergence, the differences between consecutive cumulative means are contrasted to a tolerance threshold. Thereby, when the cumulative means oscillate less than this threshold over a range of simulations, this can be denoted as the convergence of the algorithm. In this regard, Table 3 shows the simulations required considering several tolerance thresholds (from 0.01 to 0.10%) and all four FS algorithms for $k = 1$, over 50 consecutive runs. Considering a tolerance threshold of 0.03%, all algorithms converge around iteration 400. Hence, 500 runs (cross-validations) have been performed in the experimental work.

Table 3 Convergence of the algorithms for different tolerance thresholds ($k = 1$)

Tolerance (%)	MI_1L	mRMR_1L	MI_2L	mRMR_2L
0.10	147	138	137	158
0.09	158	141	137	158
0.08	191	175	202	158
0.07	217	180	216	159
0.06	283	204	279	222
0.05	346	213	320	267
0.04	346	288	361	269
0.03	406	414	407	413
0.02	0	0	0	0
0.01	0	0	0	0

4.1.2 Influence of K value in the accuracy of the algorithms

Since the best choice of k depends upon the data and the specific application, an experiment will evaluate the performance of the estimation model when using different k values for the four FS algorithms. The performance is evaluated by means of the MMRE accuracy indicator. In general, only the most similar cases are selected, suggesting a small value for k. In this study, we have used 1 to 4 nearest neighbors ($1 \leq k \leq 4$).

Table 4 shows the MMRE indicator for different values of k and considering the four FS algorithms over 500 observations. The best MMRE results are obtained for $k = 1$. This result is also obtained in (Auer et al. 2006; Chiu and Huang 2007; Liu et al. 2014). The FS algorithms mRMR_2L and MI_2L present quite similar behavior and better performance than mRMR_1L and MI_1L. Hereafter, the value of k is fixed to 1.

4.1.3 Accuracy of the Feature Selection algorithms

For $k = 1$, the mean and variance values of MMRE for the FS algorithms are MI_1L (1.4197, 0.0044), mRMR_1L (1.4174, 0.0043), MI_2L (1.3982, 0.0043), mRMR_2L (1.3956, 0.0040), GFS (1.3516, 0.0030). Figure 3 depicts MMRE notched box plots considering the proposed four FS algorithms for 500 cross-validations. The resulting notched box plots do not suggest that medians are significantly different. Anyway, it is important to check if these four means are significantly different from one another using a Wilcoxon rank test with a confidence level of 0.95 (Keung et al. 2012). This test does not assume that the difference between the samples is normally distributed.

The mean of MMRE values corresponding to FS algorithm MI_1L is significantly higher than the means corresponding to MI_2L and mRMR_2L. That also applies to the mean of MMRE values corresponding to mRMR_1L which is significantly higher than the mean corresponding to MI_2L and mRMR_2L. However, algorithms MI_1L and mRMR_1L are not significantly different; similarly, MI_2L compared to mRMR_2L. These results are consistent with Table 4. In sum, the FS algorithms mRMR_2L and MI_2L present quite similar behavior and better performance than mRMR_1L and MI_1L. Hence, it is relevant to consider two different lists of features (continuous and categorical) following Doquire and Verleysen standpoint (Doquire and Verleysen 2011). Though, regarding the usefulness of including redundancy in these algorithms, the results are non-conclusive. Finally, when comparing the aforementioned means with GFS mean, the latter presents a significantly lower value.

Besides the prediction accuracy, the computational cost is also considered in this study. The FS algorithms are tested on a INTEL Xeon E5-2620 @ 2.40 GHz with 2Gb of RAM. Table 5 presents the average and standard deviation of the execution times in seconds over 500 observations for each algorithm when $k = 1$.

Table 4 Cumulative means of MMRE

K	MI_1L	mRMR_1L	MI_2L	mRMR_2L
1	1.4197	1.4174	1.3982	1.3956
2	1.4958	1.4939	1.4730	1.4740
3	1.5929	1.5906	1.5760	1.5756
4	1.6806	1.6796	1.6645	1.6622

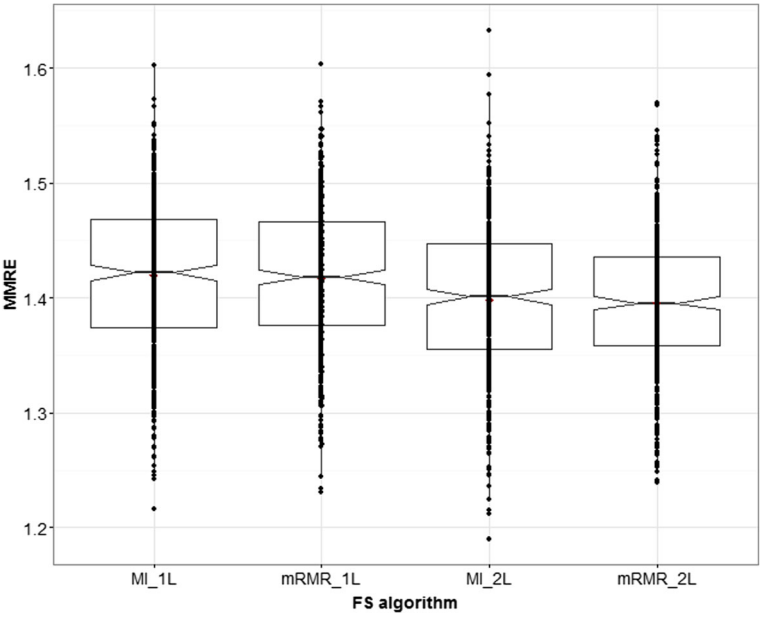

Fig. 3 Box plots of the accuracy (MMRE) regarding FS algorithms

Wrappers are usually criticized because they require massive amounts of computation; nevertheless, greedy search strategies seem to be particularly computationally advantageous and robust against overfitting (Guyon and Elisseeff 2003). In this regard, Table 5 shows that the hybrid algorithms where features are first filtered outperform GFS in terms of computational efficiency. In sum, the use of one or two ordered lists leads to an improvement of the running time (59.9% in the case of 1L–based algorithms and 38.6% in the case of 2L–based algorithms) without sacrificing too much prediction performance (4.9 and 3.3%, respectively). Finally, Table 5 also shows that the running time is not influenced by using MI or mRMR. The algorithms that use two lists are 53.3% more costly than those that use one list. Besides, the computational cost presents more variability when using two lists.

4.2 Analysis of selected features

4.2.1 Mutual information and redundancy of the independent variables

Figure 4 shows the MI between the different independent variables and the dependent variable considering the whole dataset. These features are sorted in descending order, which will guide FS in the algorithms MI_L1 and MI_L2. Indeed, this diagram provides useful information for

Table 5 Running times of the algorithms

Algorithm	Mean (seconds)	Standard deviation
MI_1L	59.67	0.951
mRMR_1L	59.65	0.964
MI_2L	91.34	11.348
mRMR_2L	91.55	11.393
GFS	148.83	37.950

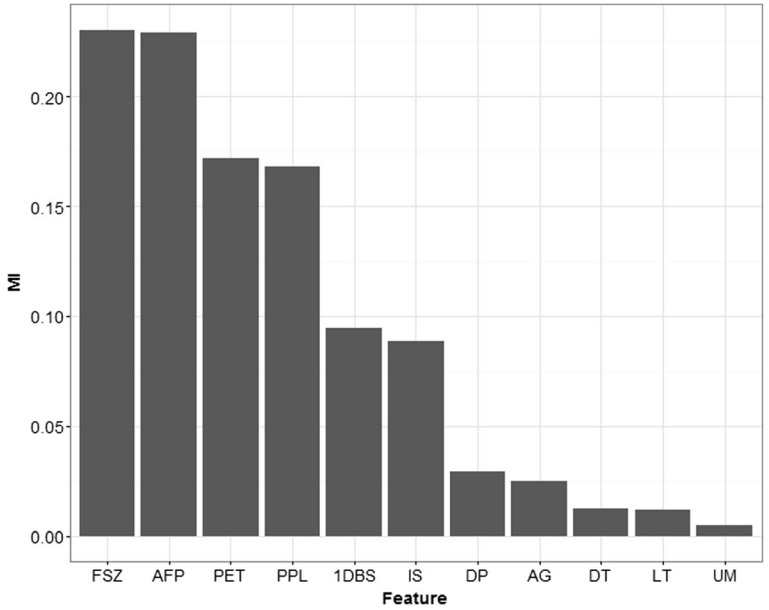

Fig. 4 Mutual information of the independent variables

diagnosing later the feature extraction of both algorithms. In Fig. 4, it is apparent that the features FSZ and AFP have the highest MI with the effort feature, followed by PET and PPL. 1DBS and IS are located in the middle, while the rest of features have very low MI (<0.05).

At this point, it is necessary to analyze the order that will guide the selection of algorithms mRMR_L1 and mRMR_L2. This order is shown in Fig. 5. Of course, the feature with the

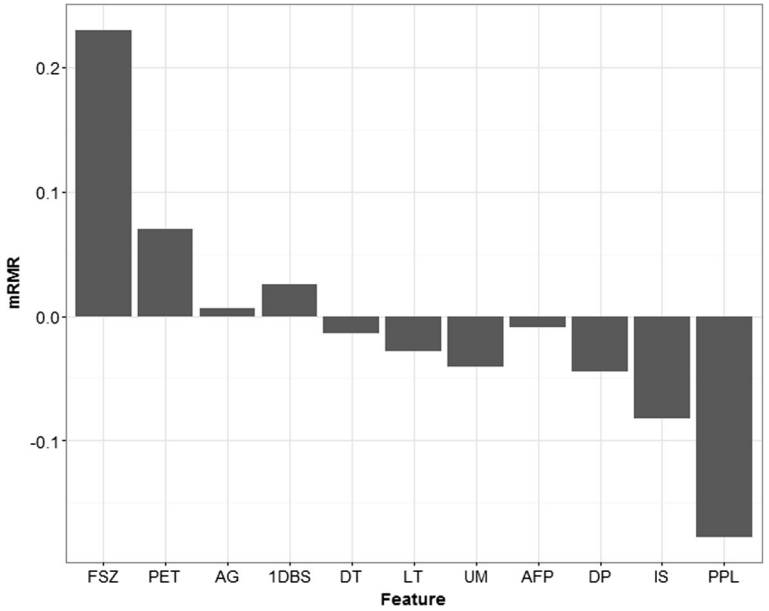

Fig. 5 mRMR of the selected features

highest MI is the first selected, i.e., FSZ. As seen in the plot, the selection of FSZ provokes that AFP (with the second highest MI value) moves to the eighth position, because of the redundancy between them. Thus, PET is selected in the second position, moving PPL to the last position. Surprisingly, AG appears in the third position due to its low dependence on the previously selected features. Then, 1DBS is selected pushing IS to the penultimate position. Note that Fig. 5 shows the selection order according to mRMR values. However, this does not imply a strictly decreasing order in mRMR values, since no normalization has been applied to the redundancy summation (Estévez et al. 2009; Vinh et al. 2010).

4.2.2 Number of selected features depending on algorithms

The number of features selected by each algorithm is analyzed in this section. Since 500 runs have been performed, Fig. 6 depicts the number of times each algorithm selects a specified number of features. It can be seen that the mode is 3 in all cases.

Table 6 shows the mean, standard deviation, and Huber's M of the number of features selected by each FS algorithm, including GFS. Huber's M is a robust statistic to summarize results when the underlying distribution is roughly normal but there are a small proportion of outliers or heavy tails.

It appears to be that the methods that use two lists employ fewer features to construct the models than those that only use one list. Using a Wilcoxon rank test with a confidence level of 0.95, all means have been proven to be significantly different from one another. In general, the four FS algorithms may lead to SDEE models that use less independent features than the average obtained (6.2 features) in (González-Ladrón-de-Guevara et al. 2016) from a total of 107 proposed effort estimation models. Anyway, GFS models even use less variables. In fact,

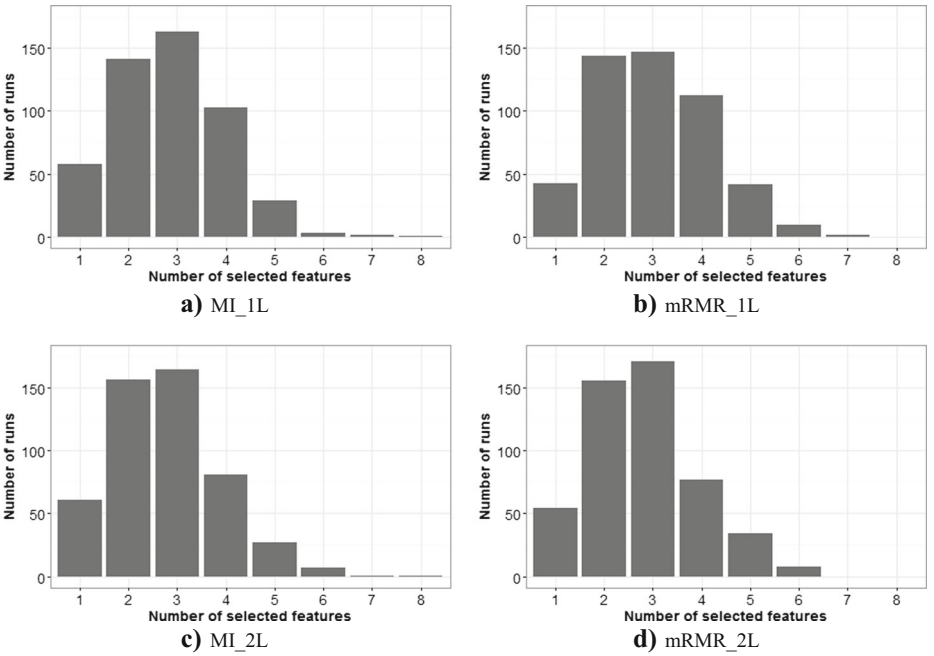

Fig. 6 Distribution of the number of selected features for each FS algorithm

Table 6 Number of features selected by each FS algorithm

Algorithm	Mean	Huber's M	Standard deviation
MI_1L	2.852	2.833	1.154
mRMR_1L	3.008	2.985	1.200
MI_2L	2.772	2.746	1.157
mRMR_2L	2.810	2.794	1.132
GFS	1.684	1.605	0.826

when the selection is not guided, all the remaining features come into play in each loop, making possible to obtain smaller models. In the next section, the preference of usage of the features is analyzed considering all algorithms.

4.2.3 Preference of usage of selected features depending on algorithms

The preference of usage of the features is analyzed in detail considering the algorithms mRMR_1L and mRMR_2L. This way, Tables 7 and 8 respectively present these results and, later in this section, a summary with the positions of the features for all algorithms is presented in Table 9.

Table 7 presents a matrix of the frequencies of usage of the independent variables, with the detail of the positions in which they have been selected by algorithm mRMR_1L over 500 runs ($k = 1$). The first column (1) shows the number of times that each feature has been selected in the first position, the second column (2) the number of times each feature has been selected in the second position, and so on for the rest of columns until the last possible position (11 in this case). The twelfth column (12) indicates the number of times each of the features has not been selected by the algorithm.

Finally, *WP* is a preference indicator of the usage of each feature that considers either the number of times this feature has been selected and the position order in which it has been selected. Provided that 11 is the total number of the independent variables, the range of *WP* is between 1 and 12. If a feature reaches $WP = 1$, it denotes that this feature has always (i.e., in all runs) been selected in the first position. On the other side, a maximum value of 12 shows that this feature has never been selected. Hence, when a feature has been selected in all 500 runs (cross-validations), it represents the selection average position of this feature. *WP* also considers the selection frequency of the feature: the less the feature is chosen, the more *WP* comes near to 12. Consequently, *WPi* shows the weighted selection position for feature *i*,

Table 7 Usage of selected features for algorithm mRMR_1L

	1	2	3	4	5	6	7	8	9	10	11	12	*WP*
FSZ	500	0	0	0	0	0	0	0	0	0	0	0	1.00
PET	0	244	0	0	0	0	0	0	0	0	0	256	7.12
AG	0	84	72	0	0	0	0	0	0	0	0	344	9.02
1DBS	0	45	67	17	0	0	0	0	0	0	0	371	9.62
DT	0	23	47	27	2	0	0	0	0	0	0	401	10.23
AFP	0	18	18	28	12	2	0	0	0	0	0	422	10.68
LT	0	15	33	23	3	1	0	0	0	0	0	425	10.68
UM	0	10	26	24	9	0	1	0	0	0	0	430	10.81
DP	0	6	22	15	12	1	1	0	0	0	0	443	11.05
IS	0	6	15	17	12	2	0	0	0	0	0	448	11.15
PPL	0	6	13	15	4	6	0	0	0	0	0	456	11.28

Table 8 Usage of selected features for algorithm mRMR_2L

	1	2	3	4	5	6	7	8	9	10	11	12	*WP*
FSZ	250	85	16	2	0	0	0	0	0	0	0	147	4.48
AG	250	77	20	0	0	0	0	0	0	0	0	153	4.60
PET	0	106	47	8	1	1	0	0	0	0	0	337	8.88
1DBS	0	83	38	12	1	0	0	0	0	0	0	366	9.45
AFP	0	30	56	22	10	0	0	0	0	0	0	382	9.90
DT	0	27	46	16	4	1	0	0	0	0	0	406	10.31
LT	0	14	22	16	3	0	0	0	0	0	0	445	11.03
UM	0	9	13	11	8	1	0	0	0	0	0	458	11.29
DP	0	7	14	11	4	1	0	0	0	0	0	463	11.36
PPL	0	2	10	12	6	3	0	0	0	0	0	467	11.47
IS	0	6	8	9	5	1	0	0	0	0	0	471	11.51

considering a weight value of 12 when the feature i has not been selected. Note that the weights of each position correspond with the position itself. *WPi* can be expressed as follows:

$$WPi = \sum_{j=1}^{12} \left(Mi, j^{*} j\right) / \text{runs} \tag{7}$$

where

Mi,j is the number of times that feature i has been selected in position j
j is the position order in which a feature i can be selected
runs is the total number of runs or cross-validations (runs = 500)

Table 7 shows that the feature FSZ is used in all cases at the first position (*WP* = 1). The second preferred feature is PET that has been used in 244 out of 500 cases and always at the second position; hence, its *WP* is much higher (7.12). AG has been used only in the second (84 times) and third place (72 times). The five first features follow exactly the order shown in Fig. 5. The rest of the features have been less used and in higher positions, as noted by their *WP* values.

Similarly, Table 8 shows the results for mRMR_2L. The most preferred feature is FSZ too and the second place, with a slight difference, is for AG that is also used in 250 out of 500 cases at the first position. This categorical feature has increased its preference of usage due to the consideration of two lists, clearly overpassing the continuous feature PET. In general, there

Table 9 Feature selection ranking regarding to *WP*

	MI_1L	mRMR_1L	MI_2L	mRMR_2L	GFS
1	FSZ	FSZ	PPL	FSZ	LT
2	AFP	PET	FSZ	AG	AG
3	PET	AG	AFP	PET	IS
4	PPL	1DBS	1DBS	1DBS	AFP
5	1DBS	DT	PET	AFP	UM
6	IS	AFP	IS	DT	PPL
7	DP	LT	DP	LT	DP
8	AG	UM	AG	UM	PET
9	DT	DP	LT	DP	FS
10	UM	IS	DT	PPL	1DBS
11	LT	PPL	UM	IS	DT

is a greater participation of several features in the first positions of the models when using the algorithm mRMR_2L but the features in the last places have been less used (higher *WP* values) than in mRMR_1L.

Next, Table 9 shows the features sorted by their weighted position (WP) for each FS algorithm including GFS too. Regarding the hybrid algorithms, it clearly shows that the feature FSZ appears at the first position in three of them. However, PPL occupies the first position when using MI_2L. The five features that have been used in the first three positions are AFP, AG, FSZ, PET, and PPL.

The continuous features appear in the first positions of Table 9 when using MI_1L. In the rest of the cases (either when using two lists or taking into account redundancy between features), the categorical features improve their position. For example, PET is overpassed by several categorical features in MI_2L. It should be noted that, when considering redundancy either in the case of one list or two lists, PPL strongly diminishes its level of usage and AFP (with a strong relationship with FSZ) decreases its level of usage too. Conversely, AG notably improves its level of usage (despite providing low MI, its redundancy with FSZ and PET is also very low). PET also slightly improves its position. When considering the nature of the features, i.e., when two lists are considered, PPL in the case of MI and AG improve their positions when redundancy is taking into account. By the way, there are features that have a quite steady level of usage in all methods such as FSZ and 1DBS.

A different situation is depicted for GFS. Due to the fact that this approach uses no reference guide for feature selection, the ordered features in the last column of Table 9 are quite different from the previous ones. Apart from that, *WP* values range from 10.02 till 10.59; hence, GFS does not discriminate and orient conveniently in regard to the features to be selected. According to the WP values in Table 8, the differences in the usage of selected features are more relevant for the first five than the rest of features. This way, for the mRMR_2L algorithm, FSZ is the most used variable not only in the experiments but usually in the SDEE methods that have used ISBSG data (González-Ladrón-de-Guevara et al. 2016). Something similar occurs regarding feature AFP. However, from Table 10 where the results from (González-Ladrón-de-Guevara et al. 2016) are compared to these experimental results, variables AG, PET, and 1DBS play a more important role in mRMR_2L algorithm than DT, LT, and DP.

Particularly, AG is a categorical feature that has been selected by the mRMR_2L algorithm in 250 out of 500 runs as the first feature of the model. This feature is derived from AT in ISBSG Release 12. In fact, AG has been used 69.4% of the times while the frequency of use of AT is 21.5% in (González-Ladrón-de-Guevara et al. 2016). Similarly, 1DBS with a frequency of use of 13.1% (González-Ladrón-de-Guevara et al. 2016) has increased its share till 26.8% in the case of the mRMR_2L algorithm. This feature has undergone a significant reduction in levels during the pre-processing stage. Finally, PET has a frequency of use of 21.5% (González-Ladrón-de-Guevara et al. 2016) while it has been selected 32.6% of the runs.

Table 10 ISBSG variables most frequently used in the literature compared to mRMR_2L algorithm

Position	(González-Ladrón-de-Guevara et al. 2016)	mRMR_2L
1	FSZ, 61.7%	FSZ, 70.6%
2	DT, 57.9%	AG, 69.4%
3	LT, 53.3%	PET, 32.6%
4	DP, 52.3%	1DBS, 26.8%
5	AFP, 28%	AFP, 23.6%

PET is the third continuous variable ordered by mutual information (Fig. 4) and the second ordered by mRMR (Fig. 5). It should be noted that from the selected ISBSG features analyzed in this paper, only FSZ, AFP, and PET are continuous and all three features get promoted by the mRMR_2L algorithm. Hence, it may be interesting to favor the usage of AG, PET, and 1DBS with preference over DT, LT, and DP.

5 Threats to validity

Several threats to validity that may affect the ability to obtain reliable conclusions have been identified. To begin with, the re-categorization of two independent features, even if irrelevant for one of them (PPL), has been described in detail. Besides, just one learner has been used in this experiment. This may be considered as a threat to validity. Nonetheless, CBR is quite sensitive to FS and is usually used to test FS algorithms. Moreover, CBR could be sensitive to the choice of distance metric. For this reason, the Gower distance has been used; allowing continuous and categorical variables to be considered in the same model and could take into account the presence of missing data.

Another threat to validity is related to the accuracy measures. As Shepperd and MacDonell reported (Shepperd and MacDonell 2012), certain measures for performance prediction like MMRE or PRED are not the most appropriate. Therefore, measures based on residuals could also be used. To mitigate the bias of this threat, a significant number of cross-validations have been performed.

Finally, some other factors are relevant when comparing FS algorithms. According to Shepperd and MacDonell (2012), one of them is the ease of use. In this regard, we have included the computational cost of the procedures and the average number of selected features. Furthermore, to avoid inconsistent results, the proposed FS algorithms were compared against a baseline.

6 Conclusions and future work

Existing FS approaches are mainly designed for classification problems with categorical or continuous features. However, the data collected in SDEE include both categorical and continuous features. There are different approaches to deal with mixed features in FS methods, the most common approach being to convert the problem into a discrete or continuous one.

This paper aims to address the problem of selecting the most relevant features from ISBSG dataset to be used in SDEE. To deal with mixed features, the approach of two separated lists proposed in (Doquire and Verleysen 2011) is followed, and provided that the continuous features have been previously discretized, the ranking of both continuous and categorical features is based on the same MI measure. The ranked list is then split into two for the continuous and categorical features and recombined according to the accuracy of a CBR model.

The experimental work performed is based on ISBSG Release 12 which includes 6006 projects and 126 features. After the filtering process, 1884 projects are remaining. This study focuses on the 20 ISBSG features most frequently used as independent variables in effort estimation models according to (González-Ladrón-de-Guevara et al. 2016). NWEL1 has been used as dependent variable to ensure maximum consistency (González-Ladrón-de-Guevara et al. 2016). Application Type and Organization Type have been replaced by AG and IS suggested in

Release 12. Eight features with missing values larger than 60% have been excluded. Resource Level has been also excluded because known development team effort is required as one of the selection criteria. Finally, projects with any missing value in the independent variables were discarded, resulting in a complete dataset with 621 projects and 12 features.

Four algorithms that include aspects of filter and wrapper approaches are considered. In all four, MI is used as a measure of both the relevance of a feature and its redundancy. Also, the mean of the *k*-nearest neighbors is used to estimate software project costs. The results of the experimental work have been obtained over 500 cross-validations for accuracy purposes. The best MMRE results are obtained for $k = 1$.

The performance of the proposed hybrid algorithms is compared between them and against a GFS wrapper approach. The hybrid algorithms improve the computational efficiency of GFS. Due to the fact that the selection is not guided by a list, GFS improves slightly the accuracy and presents models with a reduced number of variables in average. However, this is detrimental to its ability to generalize and guide the features to be selected.

Concerning the hybrid approach, FS algorithms that discriminate between continuous and categorical features (mRMR_2L and MI_2L) present quite similar behavior and better performance than those that do not discriminate (mRMR_1L and MI_1L). In fact, the mean of MMRE values corresponding to FS algorithm MI_1L is significantly higher than the means corresponding to MI_2L and mRMR_2L. The mean of MMRE values corresponding to mRMR_1L is also significantly higher than the means corresponding to MI_2L and mRMR_2L. However, algorithms MI_1L and mRMR_1L are not significantly different; similarly, MI_2L compared to mRMR_2L. Hence, it is relevant to consider two different lists of features (continuous and categorical) following Doquire and Verleysen standpoint (Doquire and Verleysen 2011). Though, regarding the usefulness of including redundancy in these algorithms, the results are non-conclusive.

Regarding the selected features in the experiments, FSZ is the most frequently selected feature when using MI_1L, mRMR_1L, and mRMR_2L. However, PPL occupies the first position when using MI_2L. The five features that have been used in the first three positions are APF, AG, FSZ, PT, and PPL. The continuous features appear in the first positions when using MI_1L. In the rest of the cases (either when using two lists or taking into account redundancy between features), the categorical features improve their position. Moreover, FSZ is not only the most used variable in the experiments but also in the SDEE methods that have used ISBSG data. The same happens with AFP. However, while variables DT, LT, and DP usually conform the group of the five most used variables, the experimental results suggest that the usage of variables AG, PET, and 1DBS should be fostered.

Again, these results denote the convenience of discriminating between continuous and categorical features. In order to select the most relevant features from ISBSG dataset to be used in SDEE, it is convenient to deepen research in this FS area. In particular, the performance could be assessed when using other FS standpoints such as INMIFS (Chandrashekar and Sahin 2014), hierarchical FS, or other applications of CBR methodology. For further work, alternative measures of performance prediction can be used, i.e., measures based on residuals such as Standardised Accuracy (Shepperd and MacDonell 2012) or Exact Mean Absolute Error of Baseline Predictor (Langdon et al. 2016). It should also be interesting to deal with missing data by applying some imputation techniques; hence, the working dataset will be larger and some other variables could be included in the selection set. Finally, it can be valuable to consider a bootstrap approach, as a complement to cross-validation, searching for obtaining robust statistics.

References

Angelis, L., & Stamelos, I. (2000). A simulation tool for efficient analogy based cost estimation. *Empirical Software Engineering, 5*(1), 35–68. https://doi.org/10.1023/A:1009897800559.

Auer, M., Trendowicz, A., Graser, B., Haunschmid, E., & Biffl, S. (2006). Optimal project feature weights in analogy-based cost estimation: improvement and limitations. *Software Engineering, IEEE Transactions on, 32*(2), 83–92.

Awada, W., Khoshgoftaar, T. M., Dittman, D., Wald, R., Napolitano, A. (2012). A review of the stability of feature selection techniques for bioinformatics data. In 2012 I.E. 13th International Conference on Information Reuse and Integration (IRI) (pp. 356–363). Presented at the 2012 I.E. 13th International Conference on Information Reuse and Integration (IRI). https://doi.org/10.1109/IRI.2012.6303031.

Battiti, R. (1994). Using mutual information for selecting features in supervised neural net learning. *Neural Networks, IEEE Transactions, 5*(4), 537–550.

Bennasar, M., Hicks, Y., & Setchi, R. (2015). Feature selection using joint mutual information maximisation. *Expert Systems with Applications, 42*(22), 8520–8532. https://doi.org/10.1016/j.eswa.2015.07.007.

Bibi, S., Tsoumakas, G., Stamelos, I., & Vlahavas, I. (2008). Regression via classification applied on software defect estimation. *Expert Systems with Applications, 34*(3), 2091–2101. https://doi.org/10.1016/j.eswa.2007.02.012.

Chandrashekar, G., & Sahin, F. (2014). A survey on feature selection methods. *Computers & Electrical Engineering, 40*(1), 16–28.

Chatzipetrou, P., Papatheocharous, E., Angelis, L., Andreou, A. S. (2012). An investigation of software effort phase distribution using compositional data analysis. In 2012 38th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA) (pp. 367–375). Presented at the 2012 38th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA). https://doi.org/10.1109/SEAA.2012.50.

Chen, Z., Menzies, T., Port, D., & Boehm, B. (2005). Feature subset selection can improve software cost estimation accuracy. In *Proceedings of the 2005 workshop on predictor models in software engineering* (pp. 1–6). New York: ACM. https://doi.org/10.1145/1082983.1083171.

Chiu, N.-H., & Huang, S.-J. (2007). The adjusted analogy-based software effort estimation based on similarity distances. *Journal of Systems and Software, 80*(4), 628–640.

Dash, M., & Liu, H. (2003). Consistency-based search in feature selection. *Artificial Intelligence, 151*(1), 155–176.

Dejaeger, K., Verbeke, W., Martens, D., & Baesens, B. (2012). Data mining techniques for software effort estimation: a comparative study. *Software Engineering, IEEE Transactions on, 38*(2), 375–397. https://doi.org/10.1109/TSE.2011.55.

Deng, K., & MacDonell, S. G. (2008). Maximising data retention from the ISBSG repository. In *Proceedings of the 12th international conference on evaluation and assessment in software engineering* (pp. 21–30). Swinton: British Computer Society http://dl.acm.org/citation.cfm?id=2227115.2227118. Accessed 21 Jan 2014.

Doquire, G., & Verleysen, M. (2011). An hybrid approach to feature selection for mixed categorical and continuous data. In International Conference on Knowledge Discovery and Information Retrieval. http://hdl.handle.net/2078.1/90765. Accessed 2 Nov 2015.

Dudani, S. A. (1976). The distance-weighted k-nearest-neighbor rule. *IEEE Transactions on Systems, Man and Cybernetics, SMC, 6*(4), 325–327. https://doi.org/10.1109/TSMC.1976.5408784.

Estévez, P. A., Tesmer, M., Perez, C. A., & Zurada, J. M. (2009). Normalized mutual information feature selection. *IEEE Transactions on Neural Networks, 20*(2), 189–201. https://doi.org/10.1109/TNN.2008.2005601.

Fayyad, U.M., & Irani, K.B. (1993). Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning. In Proceedings of the International Joint Conference on Uncertainty in AI (pp. 1022–1027). Presented at the International Joint Conference on Uncertainty in AI. https://www.researchgate.net/publication/220815890_Multi-Interval_Discretization_of_Continuous-Valued_Attributes_for_Classification_Learning. Accessed 22 June 2016.

Fernández-Diego, M., & González-Ladrón-de-Guevara, F. (2014). Potential and limitations of the ISBSG dataset in enhancing software engineering research: a mapping review. *Information and Software Technology, 56*(6), 527–544. https://doi.org/10.1016/j.infsof.2014.01.003.

Ferreira, A., & Figueiredo, M. (2011). Unsupervised joint feature discretization and selection. In J. Vitrià, J. M. Sanches, & M. Hernández (Eds.), *Pattern recognition and image analysis* (Vol. 6669, pp. 200–207). Berlin, Heidelberg: Springer Berlin Heidelberg http://link.springer.com/10.1007/978-3-642-21257-4_25. Accessed 4 Mar 2016.

Fleuret, F. (2004). Fast binary feature selection with conditional mutual information. *Journal of Machine Learning Research, 5*, 1531–1555.

González-Ladrón-de-Guevara, F., Fernández-Diego, M., & Lokan, C. (2016). The usage of ISBSG data fields in software effort estimation: a systematic mapping study. *Journal of Systems and Software, 113*, 188–215. https://doi.org/10.1016/j.jss.2015.11.040.

Gupta, P., Jain, S., & Jain, A. (2014). A review of fast clustering-based feature subset selection algorithm. *International Journal of Scientific & Technology Research, 3*(11), 86–91.

Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. *The Journal of Machine Learning Research, 3*, 1157–1182.

Hall, M. A., & Holmes, G. (2003). Benchmarking attribute selection techniques for discrete class data mining. *IEEE Transactions on Knowledge and Data Engineering, 15*(6), 1437–1447. https://doi.org/10.1109/TKDE.2003.1245283.

Hausser, J., & Strimmer, K. (2009). Entropy inference and the James-Stein estimator, with application to nonlinear gene association networks. *Journal of Machine Learning Research, 10*(Jul), 1469–1484.

Hill, P. (2010). Practical software project estimation: a toolkit for estimating software development effort & duration. McGraw Hill Professional.

Hsu, H.-H., Hsieh, C.-W., & Lu, M.-D. (2011). Hybrid feature selection by combining filters and wrappers. *Expert Systems with Applications, 38*(7), 8144–8150.

Huang, S.-J., & Chiu, N.-H. (2006). Optimization of analogy weights by genetic algorithm for software effort estimation. *Information and Software Technology, 48*(11), 1034–1045. https://doi.org/10.1016/j.infsof.2005.12.020.

Huang, S.-J., Chiu, N.-H., & Liu, Y.-J. (2008). A comparative evaluation on the accuracies of software effort estimates from clustered data. *Information and Software Technology, 50*(9–10), 879–888. https://doi.org/10.1016/j.infsof.2008.02.005.

Huang, J., Li, Y.-F., & Xie, M. (2015). An empirical analysis of data preprocessing for machine learning-based software cost estimation. *Information and Software Technology, 67*, 108–127. https://doi.org/10.1016/j.infsof.2015.07.004.

ISBSG. (2013a). ISBSG Dataset Release 12. *ISBSG*. http://isbsg.org/. Accessed 1 Mar 2016.

ISBSG. (2013b). ISBSG Guidelines Release 12.

ISBSG. (2013c). ISBSG Data Demographics Release 12.

Jeffery, R., Ruhe, M., Wieczorek, I. (2001). Using public domain metrics to estimate software development effort. In Software Metrics Symposium, 2001. METRICS 2001. Proceedings. Seventh International (pp. 16–27). https://doi.org/10.1109/METRIC.2001.915512.

Jiang, Z., & Comstock, C. (2007). The factors significant to software development productivity. In C. Ardil (Ed.), Proceedings of World Academy of Science, Engineering and Technology, Vol 19 (Vol. 19, pp. 160–164). Presented at the Conference of the World-Academy-of-Science-Engineering-and-Technology, Bangkok: World Acad Sci, Eng & Tech-Waset.

Jørgensen, M., Indahl, U., & Sjøberg, D. (2003). Software effort estimation by analogy and 'regression toward the mean'. *Journal of Systems and Software, 68*(3), 253–262. https://doi.org/10.1016/S0164-1212(03)00066-9.

Kabir, M. M., Shahjahan, M., & Murase, K. (2011). A new local search based hybrid genetic algorithm for feature selection. *Neurocomputing, 74*(17), 2914–2928.

Kadoda, G., Cartwright, M., Chen, L., Shepperd, M. (2000). Experiences using case-based reasoning to predict software project effort. In EASE 2000 (pp. 2–3). Presented at the EASE 2000, Staffordshire, UK.

Keung, J., Kocaguneli, E., & Menzies, T. (2012). Finding conclusion stability for selecting the best effort predictor in software effort estimation. *Automated Software Engineering, 20*(4), 543–567. https://doi.org/10.1007/s10515-012-0108-5.

Kirsopp, C., Shepperd, M. J., Hart, J. (2002). Search heuristics, case-based reasoning and software project effort prediction. In Proceedings of the Genetic and Evolutionary Computation Conference (pp. 9–13). New York, USA. http://v-scheiner.brunel.ac.uk/handle/2438/1554. Accessed 27 Jan 2016.

Kohavi, R., & John, G. H. (1997). Wrappers for feature subset selection. *Artificial Intelligence, 97*(1–2), 273–324. https://doi.org/10.1016/S0004-3702(97)00043-X.

Kwak, N., & Choi, C.-H. (2002). Input feature selection for classification problems. *IEEE Transactions on Neural Networks, 13*(1), 143–159. https://doi.org/10.1109/72.977291.

Langdon, W. B., Dolado, J., Sarro, F., & Harman, M. (2016). Exact mean absolute error of baseline predictor, MARP0. *Information and Software Technology, 73*, 16–18. https://doi.org/10.1016/j.infsof.2016.01.003.

Li, Y. F., Xie, M., & Goh, T. N. (2009). A study of mutual information based feature selection for case based reasoning in software cost estimation. *Expert Systems with Applications, 36*(3), 5921–5931.

Liu, H., & Motoda, H. (2012). Feature selection for knowledge discovery and data mining (Vol. 454). Springer Science & Business Media. https://books.google.es/books?hl=en&lr=&id=aaDbBwAAQBAJ&oi=fnd&pg=PP10&dq=Feature+selection+for+knowledge+discovery+and+data+mining&ots=iuMhcWZGcf&sig=KlmNEIcsBdDVs-m1HUuICfpYZiM. Accessed 25 Jan 2016.

Liu, H., & Yu, L. (2005). Toward integrating feature selection algorithms for classification and clustering. *IEEE Transactions on Knowledge and Data Engineering, 17*(4), 491–502. https://doi.org/10.1109/TKDE.2005.66.

Liu, H., Wei, R., & Jiang, G. (2013). A hybrid feature selection scheme for mixed attributes data. *Computational and Applied Mathematics, 32*(1), 145–161. https://doi.org/10.1007/s40314-013-0019-5.

Liu, Q., Wang, J., Xiao, J., Zhu, H. (2014). Mutual information based feature selection for symbolic interval data. In International Conference on Software Intelligence Technologies and Applications International Conference on Frontiers of Internet of Things 2014 (pp. 62–69). Presented at the International Conference on Software Intelligence Technologies and Applications International Conference on Frontiers of Internet of Things 2014. https://doi.org/10.1049/cp.2014.1537.

Lokan, C. (2005). What should you optimize when building an estimation model? In Software Metrics, 2005. 11th IEEE International Symposium (pp. 1–10). https://doi.org/10.1109/METRICS.2005.55.

Lokan, C., & Mendes, E. (2009a). Investigating the use of chronological split for software effort estimation. *Software, IET, 3*(5), 422–434. https://doi.org/10.1049/iet-sen.2008.0107.

Lokan, C., & Mendes, E. (2009b). Applying moving windows to software effort estimation. In *Proceedings of the 2009 3rd international symposium on empirical software engineering and measurement* (pp. 111–122). Washington, DC: IEEE Computer Society. https://doi.org/10.1109/ESEM.2009.5316019.

Lokan, C., & Mendes, E. (2012). Investigating the use of duration-based moving windows to improve software effort prediction. In Software Engineering Conference (APSEC), 2012 19th Asia-Pacific (Vol. 1, pp. 818–827). Presented at the Software Engineering Conference (APSEC), 2012 19th Asia-Pacific. https://doi.org/10.1109/APSEC.2012.74.

Lustgarten, J.L., Visweswaran, S., Grover, H., Gopalakrishnan, V. (2008). An evaluation of discretization methods for learning rules from biomedical datasets. In BIOCOMP (pp. 527–532).

Mandal, M., & Mukhopadhyay, A. (2013). An improved minimum redundancy maximum relevance approach for feature selection in gene expression data. *Procedia Technology, 10*, 20–27. https://doi.org/10.1016/j.protcy.2013.12.332.

Mendes, E., Watson, I., Triggs, C., Mosley, N., & Counsell, S. (2003). A comparative study of cost estimation models for web hypermedia applications. *Empirical Software Engineering, 8*(2), 163–196.

Mendes, E., Lokan, C., Harrison, R., Triggs, C. (2005). A replicated comparison of cross-company and within-company effort estimation models using the ISBSG database. In Software Metrics, 2005. 11th IEEE International Symposium (pp. 1–10). https://doi.org/10.1109/METRICS.2005.4.

Moses, J., Farrow, M., Parrington, N., & Smith, P. (2006). A productivity benchmarking case study using Bayesian credible intervals. *Software Quality Journal, 14*(1), 37–52. https://doi.org/10.1007/s11219-006-6000-4.

Núñez, H., Sànchez-Marrè, M., Cortés, U., Comas, J., Martínez, M., Rodríguez-Roda, I., & Poch, M. (2004). A comparative study on the use of similarity measures in case-based reasoning to improve the classification of environmental system situations. *Environmental Modelling & Software, 19*(9), 809–819. https://doi.org/10.1016/j.envsoft.2003.03.003.

Oh, I.-S., Lee, J.-S., & Moon, B.-R. (2004). Hybrid genetic algorithms for feature selection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on, 26*(11), 1424–1437.

Peng, H., Long, F., & Ding, C. (2005). Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 27*(8), 1226–1238. https://doi.org/10.1109/TPAMI.2005.159.

R Core Team. (2015). *R: A language and environment for statistical computing*. Vienna: R Foundation for Statistical Computing https://www.R-project.org/.

Romanski, P., & Kotthoff, L. (2014). FSelector: Selecting attributes. R package version 0.20. https://CRAN.R-project.org/package=FSelector.

Shannon, C. E. (1949). *The mathematical theory of communication*. Urbana: University of Illinois Press.

Shepperd, M., & MacDonell, S. (2012). Evaluating prediction systems in software project estimation. *Information and Software Technology, 54*(8), 820–827.

Shepperd, M., & Schofield, C. (1997). Estimating software project effort using analogies. *Software Engineering, IEEE Transactions on, 23*(11), 736–743.

Somol, P., Pudil, P., & Kittler, J. (2004). Fast branch & bound algorithms for optimal feature selection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on, 26*(7), 900–912.

Song, Q., & Shepperd, M. (2007). A new imputation method for small software project data sets. *Journal of Systems and Software, 80*(1), 51–62.

Top, O. O., Ozkan, B., Nabi, M., Demirors, O. (2011). Internal and External Software Benchmark Repository Utilization for Effort Estimation. In Software Measurement, 2011 Joint Conference of the 21st Int'l Workshop on and 6th Int'l Conference on Software Process and Product Measurement (IWSM-MENSURA) (pp. 302–307). https://doi.org/10.1109/IWSM-MENSURA.2011.41.

Vinh, L.T., Thang, N.D., Lee, Y.-K. (2010). An improved maximum relevance and minimum redundancy feature selection algorithm based on normalized mutual information. In 2010 10th IEEE/IPSJ International Symposium on Applications and the Internet (SAINT) (pp. 395–398). Presented at the 2010 10th IEEE/IPSJ International Symposium on Applications and the Internet (SAINT). https://doi.org/10.1109/SAINT.2010.50.

Witten, I.H., Frank, E., Hall, M.A., Pal, C.J. (2011). Data mining: Practical machine learning tools and techniques. Morgan Kaufmann.

Marta Fernández-Diego received her European PhD in Electronics and Telecommunications Engineering from Lille University of Science and Technology (France) in 2001. For several years, she belonged to a software development team for mobile phone applications in an international information technology services company. She is currently a lecturer in the Department of Business Organisation at Universitat Politècnica de València (Spain), where she teaches in the School of Informatics. Her research interests include empirical software engineering, software effort estimation, business analytics, and project risk management.

Fernando González-Ladrón-de-Guevara has worked at several universities and IT companies across Europe and Latin America. At the moment, he is an Associate Professor in the Telecommunications Engineering School at Universitat Politècnica de València (UPV). With a Ph.D. in Industrial Engineering since 2001, he has published articles in well-known international journals and regularly participates in several organizing committees of national and international conferences. His research interests include crowdsourcing, ERP systems, and software engineering. He has participated in 27 research projects and contracts with different organizations, being responsible of seven of them.