The Journal of Systems and Software 81 (2008) 2361–2370

Contents lists available at ScienceDirect

The Journal of Systems and Software

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

Can *k*-NN imputation improve the performance of C4.5 with small software project data sets? A comparative evaluation

Qinbao Song [a,*], Martin Shepperd [b], Xiangru Chen [a], Jun Liu [c]

[a] Department of Computer Science & Technology, Xi'an Jiaotong University, 28 Xian-Ning West Road, Xi'an, Shaanxi, 710049, China
[b] School of IS, Computing & Maths, Brunel University, Uxbridge, UB8 3PH, United Kingdom
[c] Shaanxi Electric Power Training Center for the Staff Members, 21 Dian-Chang East Road, Xi'an, Shaanxi, 710038, China

ARTICLE INFO

Article history:
Received 25 January 2007
Received in revised form 6 May 2008
Accepted 6 May 2008
Available online 17 May 2008

Keywords:
Missing data
Missing data toleration
C4.5
Data imputation
Software project cost prediction

ABSTRACT

Missing data is a widespread problem that can affect the ability to use data to construct effective prediction systems. We investigate a common machine learning technique that can tolerate missing values, namely C4.5, to predict cost using six real world software project databases. We analyze the predictive performance after using the *k*-NN missing data imputation technique to see if it is better to tolerate missing data or to try to impute missing values and then apply the C4.5 algorithm. For the investigation, we simulated three missingness mechanisms, three missing data patterns, and five missing data percentages. We found that the *k*-NN imputation can improve the prediction accuracy of C4.5. At the same time, both C4.5 and *k*-NN are little affected by the missingness mechanism, but that the missing data pattern and the missing data percentage have a strong negative impact upon prediction (or imputation) accuracy particularly if the missing data percentage exceeds 40%.

1. Introduction

Missing data is a widespread problem that can affect the ability to use data to construct effective prediction systems because few statistical or machine learning techniques can cope with such situations. Until recently the usual approach has been to exclude data that contains one or more missing values. This is often referred to as missing data ignoring technique, specifically, case or list-wise deletion. Although it is a simple approach, it suffers from two substantial problems. First, it leads to the inefficient use of already scarce data. Second, case deletion can introduce substantial bias into a data set unless the values are missing completely at random. Therefore, missing data toleration techniques and missing data imputation techniques (see Section 2.3 for details) have been proposed as alternative approaches to deal with missing values.

Unfortunately, no one technique is consistently effective. Therefore, a number of papers on the evaluation of missing data techniques has been published (e.g. Little and Rubin, 1989; Brockmeier et al., 1998; Feelders, 1999; Grzymala-Busse and Hu, 2000; Strike et al., 2001; Myrtveit et al., 2001; Batista and Monard, 2003; Cartwright et al., 2003). However, these works all focus on missing data ignoring techniques and missing data imputation techniques. To our knowledge, the comparison of missing data toleration techniques and missing data imputation techniques remains unexplored. So the objective of this paper is to consider this problem. As missing data toleration techniques are embedded in machine learning methods themselves and missing data imputation techniques aim to provide complete data to data analysis methods; moreover, even a better estimate for each missing value not necessarily leads to a better overall estimate for the parameters of interest (Rubin, 1996), so it is reasonable to evaluate missing data techniques by means of comparing the results of statistical or machine learning methods with original and treated data (see for example, Strike et al., 2001; Myrtveit et al., 2001; Batista and Monard, 2003). Therefore, our study is in the context of software project development cost prediction with a real world data set.

Software project development cost prediction is concerned with estimating how much cost will be consumed in order to develop a software project. It is an important and active research area but with many challenges remaining, including dealing with missing data (a common occurrence), small data sets with complex interactions between features (multi-collinearity) and rapid obsolescence of data due to technology changes. Thus a large number of papers has been published (Walkerden and Jeffery, 1997) on this topic. We classify the published methods into three categories: parametric model based methods, expert judgement methods, and machine learning methods.

Expert judgement techniques such as Work Breakdown Structure (Boehm, 1981; Baird, 1989) and pairwise comparison (Miranda, 2001; Shepperd and Cartwright, 2001) based on experts'

* Corresponding author.
E-mail addresses: qbsong@mail.xjtu.edu.cn (Q. Song), martin.shepperd@brunel.ac.uk (M. Shepperd).

0164-1212/$ - see front matter © 2008 Elsevier Inc. All rights reserved.
doi:10.1016/j.jss.2008.05.008

knowledge and experience and are useful in the absence of quantified, empirical data. The limitation of these methods is that an estimate is only as good as the experts' opinions which there is usually no way to test. There are also difficulties with transparency and repeatability.

Parametric models such as SLIM (Putnam, 1982), COCOMO (Boehm, 1981) and SEER (Jensen, 1983) use pre-specified formulas to estimate software cost. Unfortunately, they tend only to perform well in their own environments (Kemerer, 1987; Kitchenham, 1992), and need local data to calibrate the model to new circumstances. But even with calibration the accuracy can be quite mixed and missing values cause considerable difficulties.

Machine learning methods (Srinivasan and Fisher, 1995; Shepperd and Schofield, 1997; Finnie and Wittig, 1997) use historical software project data sets to predict cost, and can be very accurate given appropriate training data. Unfortunately, most of them cannot work with missing data. But in practice, historical software project data sets usually contain missing values (Briand et al., 1992; Briand, 1993; Gray and MacDonnell, 1997; Angelis et al., 2001; Jeffery et al., 2001). Missing values pose a challenge to machine learning methods.

Of these three categories of cost prediction methods, the last two are impacted by missing values. Thus, dealing with missing data also is an important issue in software engineering. At the same time, in the context of software cost prediction, either ignoring techniques (Walston and Felix, 1977; Briand et al., 1999, 2000) or imputation techniques (Strike et al., 2001; Myrtveit et al., 2001; Cartwright et al., 2003; Jönsson and Wohlin, 2004; Song et al., 2005; Song and Shepperd, 2007) have been used to deal with missing values. However, ignoring techniques make small software project data sets more smaller, and most imputation techniques are based on assumptions about missingness mechanisms that are not testable. By contrast, missing data toleration techniques directly analyze data sets with missing values, and have no assumptions on data distribution and missingness mechanism. On the other hand, to our knowledge, missing data toleration techniques are not used in software cost prediction. For this reason we choose to study missing data toleration techniques in the context of software project cost prediction.

The well known missing data toleration techniques include CART (Breiman et al., 1984), CN2 (Clark and Niblett, 1989), Ltree (Gama and Brazdil, 1999), and C4.5 (Quinlan, 1993). Among these methods, CART has been used in software cost modeling (Briand and Wust, 2001; Pickard et al., 2001; Briand et al., 1999) but not to deal with missing data. However, both single imputation and multiple imputation (Rubin, 1977, 1978) tend to outperform CART (Feelders, 1999). Ltree is an oblique decision tree which follows the same strategy for handling missing values as C4.5. C4.5 is one of the best methods on treating missing values among nine approaches (Grzymala-Busse and Hu, 2000) including CN2 which fills in the missing values with the most common value, the capacity of tackling both continuous and categorical values further makes it fitting software project data sets that always contain both types of data, and it has been used to identify software modules with high cost (Selby and Porter, 1988; Tian et al., 1992) with complete data.

All the above motivates us to consider the use of C4.5 to predict project cost in the face of missing values. Specifically, we explore two possibilities. One, we can use the technique directly or two, we can use a separate imputation technique prior to applying C4.5 upon the artificially completed data set. In other words, can an imputation technique improve the performance of the missing data toleration method C4.5 when using it as a predictor?

The remainder of the paper is organized as follows. In the next section we present the basic concepts of missing data techniques. This is followed by a description of the research method we used. The results follow with concluding discussion and suggestions for further work.

2. Concepts of missing data techniques

Missing values introduce complexities to data analysis. The assumptions one makes about the missingness mechanism and the missing data pattern of missing values can affect missing data dealing with methods.

2.1. Missingness mechanisms

Missingness mechanisms are assumptions about the nature and types of missing data. Little and Rubin (2002) defined three types of missing data mechanisms: missing completely at random (MCAR), missing at random (MAR), and non-ignorable (NI).

In general, a missingness mechanism concerns whether the missingness is related to the study variables or not. This is extremely significant as it determines how difficult it may be to handle missing values and at the same time how risky it is to ignore them.

Missingness mechanisms can be described as follows (Rubin, 1976). Suppose $\mathscr{Z}$ is a data matrix that includes observed and missing data, let $\mathscr{Z}_{\text{obs}}$ be the set of observed values of $\mathscr{Z}$, $\mathscr{Z}_{\text{mis}}$ be the set of missing values of $\mathscr{Z}$, $\mathscr{R}$ be the missing data indicator matrix, i be the ith case and j the jth feature. Then

$$\mathscr{R}_{i,j} = \begin{cases} 1 & \text{if } \mathscr{Z}_{i,j} \text{ is missing,} \\ 0 & \text{if } \mathscr{Z}_{i,j} \text{ is observed.} \end{cases}$$

Missing completely at random (MCAR) indicates that the missingness is unrelated to the values of any variables, whether missing or observed. So

$$p(\mathscr{R}|\mathscr{Z}) = p(\mathscr{R}) \quad \text{for all } \mathscr{Z}.$$

MCAR is an extreme condition and from an analysts point of view, ideal. Generally you can test whether MCAR condition can be met by showing there is no difference between the distribution of the observed cases and the missing cases, this is Little's (1988), Little et al.'s (1995) multivariate test which is implemented in SYSTAT and the SPSS missing values analysis module. Unfortunately, this is hard when there are few cases as there can be a problem with Type I errors.

Non-ignorable (NI) is at the opposite end of the spectrum. It means that the missingness is non-random, it is related to the missing values, and it is not predictable from any one variable in the data set. That is

$$p(\mathscr{R}|\mathscr{Z}) \neq p(\mathscr{R}) \text{ for all } \mathscr{Z}, p(\mathscr{R}|\mathscr{Z}) \text{ depends on } \mathscr{Z}_{\text{mis}}.$$

NI is the worst case since, as the name implies, the problem cannot be avoided by a deletion technique nor by imputation techniques in general effective unless the analyst has some model of the cause of missingness. This is best illustrated by an example. Suppose software engineers are less likely to report high defect rates than low rates, perhaps for reasons of politics. Merely to ignore the incomplete values leads to a biased sample and an over optimistic view of defects. On the other hand, imputation techniques do not work well either since they attempt to exploit known values and as we have already observed this is a biased sample. Unless one has some understanding of the process and can construct explanatory models, there is little that can effectively be done with the NI missingness mechanism.

Missing at random (MAR) lies between these two extremes. It requires that the cause of the missing data is unrelated to the missing values, but may be related to the observed values of other variables. That is

$$p(\mathscr{R}|\mathscr{Z}) = p(\mathscr{R}|\mathscr{Z}_{\text{obs}}) \quad \text{for all } \mathscr{Z}_{\text{mis}}.$$

Using the same example as for NI, this would occur if smaller projects were less likely to report defect rates than larger projects. Most missing data methods assume MAR. Whether the MAR condition holds can be examined by a simple *t*-test of mean differences between the group with complete data and that with missing data (Kim and Curry, 1977; Tabachnick and Fidell, 2001). MAR is less restrictive than MCAR because MCAR is a special case of MAR. MAR and MCAR are both said to be ignorable missing data mechanisms (Rubin, 1976) and is fully explained in the context of multiple imputation in Rubin (1987).

In practice it is usually difficult to meet the MCAR assumption. MAR is an assumption that is more often, but not always tenable.

2.2. Missing data patterns

The missing data indictor matrix $\mathscr{R}$ reveals the missing data pattern. Generally, there are two types of missing data patterns, they are the *univariate pattern* and the *multivariate pattern*.

In the *univariate pattern*, only one variable contains missing values. Table 1 is an example. In Table 1, only variable x_3 contains three missing values.

In the *multivariate pattern*, more than one variable contain missing data. We can refine this pattern into two types: the *monotone pattern* and the *arbitrary pattern*.

In the *monotone pattern*, variables can be arranged so that for a set of variables $x_1, x_2, \ldots, x_n$, if x_i is missing, then so are $x_{i+1}, \ldots, x_n$. Table 2 is an example.

In the *arbitrary pattern*, missing data can occur anywhere and no special structure appears regardless how you arrange variables. Table 3 is an example.

The types of missing data patterns may affect the selection of missing data methods, because some missing data methods are sensitive to missing data patterns. For this reason we will examine different patterns in our experimental analysis.

Table 1
The univariate missing pattern

	x_1	x_2	x_3	x_4	x_5	x_6
C1	*	*	*	*	*	*
C2	*	*	*	*	*	*
C3	*	*		*	*	*
C4	*	*	?	*	*	*
C5	*	*	?	*	*	*
C6	*	*	*	*	*	*

Table 2
The monotone missing pattern

	x_1	x_2	x_3	x_4	x_5	x_6
C1	*	*	*	*	*	*
C2	*	*	*	*	*	
C3	*	*	*	*		?
C4	*	*	*		?	?
C5	*	*		?	?	?
C6	*		?	?	?	?
		?	?	?	?	?

Table 3
The arbitrary missing pattern

	x_1	x_2	x_3	x_4	x_5	x_6
C1	*	*	?	*	*	*
C2	*	*	*	*	*	
C3	*	*	*	*	*	?
C4			*	*	*	?
C5	?	?	*	*	*	*
C6	*	*	*	*	*	*
	?					

2.3. Missing data techniques

The missing data problem has been studied by researchers in many fields for more than 30 years. There are three approaches to this problem: missing data ignoring techniques, missing data toleration techniques, and missing data imputation techniques.

Missing data ignoring techniques simply delete the cases that contain missing data. Because of their simplicity, they are widely used and tend to be the default for most statistics packages, but this may not lead to the most efficient utilization of the data and incurs a bias in the data unless the values are missing completely at random. Consequently they should be used only in situations where the level of missing values is very low.

Missing data toleration techniques are the internal missing data treatment strategies, which perform analysis directly using the data sets with missing values. If the objective is not to predict the missing values, missing data toleration is a better choice. This is because any predication of missing values will incur bias thereby making prediction results doubtful.

Missing data imputation techniques refer to any strategy for filling in missing values of a data set so that standard methods can then be applied to analyze the completed data set. These techniques not only retain data in incomplete cases, but also impute values of correlated variables (Little and Rubin, 1989).

Missing data imputation techniques can be classified into the ignorable missing data imputation methods, which consist of the single imputation methods and the multiple imputation methods (Rubin, 1977, 1978), and the non-ignorable missing data imputation methods which consist of the likelihood based methods (Little, 1995) and the non-likelihood based methods (Robins, 1997). A single imputation method fills in one value for each missing value, it is more common currently than multiple imputation which replaces each missing value with several plausible values and better reflects sampling variability about the actual values. However, multiple imputation generates more than one data set, so how to integrate the multiple values induced still needs to be addressed.

The *k*-NN (*k*-nearest neighbors) imputation is a hot-deck single imputation method, it fills in missing data by taking values from other observations in the same data set. This method searches the *k*-nearest neighbors of the case with missing value(s) and replaces the missing value(s) by the mean or mode value of the corresponding feature values of the *k*-nearest neighbors. The advantages of the *k*-NN imputation are:

1. It does not require to create a predictive model for each feature with missing data.
2. It can treat both continuous and categorical values.
3. It can easily deal with cases with multiple missing values.
4. It takes into account the correlation structure of the data.

Most notably, *k*-NN has no explicit missingness mechanism assumption, this makes it practically useful. Therefore, we used the *k*-NN method to impute the simulated missing values in the data sets.

3. Feature subset selection

All features are necessary for characterizing a set of software projects, but not all of them are necessary for predicting software cost. Therefore, before predicting software cost, we must first decide which feature subset is useful for the prediction task.

Therefore, in this subsection, we introduce the feature subset selection method used by the research.

Feature subset selection (Cheung and Eisenstein, 1978; Khotanzad and Kashyap, 1987; Siedelecki and Skalansky, 1988; Jain and Zongker, 1997) (FSS) is the process of identifying and removing as much irrelevant and redundant information as possible.

The existing FSSs fall into three categories: the wrapper, filter and embedded methods. A wrapper method uses a predetermined selection algorithm to search for feature subsets, and employs a induction algorithm to evaluate them iteratively and make the final decision. The same induction algorithm will be used to induce the final target concept. This type of method can obtain high induction accuracy but inherits the limitations of the induction algorithm and is highly expensive in terms of the computational cost. A filter method filters the irrelevant features before applying an induction algorithm, it is much faster than wrappers and hence can be applied to large data sets with many features. But there is a danger that features selected by this method cannot allow an induction algorithm to reach its maximum accuracy.

By contrast, an embedded method does the feature selection inside the induction algorithm itself. C4.5 is an example. At the same time, sometimes C4.5 is also used as a filter method. Cardie (1993) used a decision tree to select feature subset for a nearest neighbor algorithm for a natural language processing task, the features that did not appear in the resulting tree were removed. The results show clearly that the quality of the subset generated by a decision tree helped the nearest neighbor algorithm to reduce its prediction error. Kubat et al. (1993) used C4.5 filtering features for use with a naïve Bayesian classifier and obtained a similar result. But Kibler and Aha (1987) reported more mixed results on two medical classification tasks.

John et al. (1994) advocated the wrapper model as a means of identifying useful feature subsets and used C4.5 as the induction engine. They tested both forward stepwise selection and backward stepwise elimination on several data sets. Their results show the improvement of prediction accuracy of C4.5 is not significant, which seems to be in line with Holte's (1993) claims. But Kohavi and John (1997) found it is hard to use C4.5 to obtain the optimal feature subset and that forward selection search can improve the accuracy of C4.5. John (1997) showed that a single irrelevant feature to the credit-approval or diabetes data sets reduced the prediction accuracy of C4.5 by 5%.

To summarize, the results of using C4.5 either as a feature subset selection method or as an induction engine with other feature subset selection method can be mixed. However, Yu and Liu's fast correlation-based filter (FCBF) (Yu and Liu, 2003) method can improve the accuracy of C4.5 (Yu and Liu, 2003). So we decided using FCBF to select the feature subset for software cost prediction via C4.5.

Unfortunately, although FCBF works well with discrete and nominal features, it cannot tackle continuous features. So before applying FCBF, we first used the Chi2 discretization algorithm (Tay and Shen, 2002) converting continuous values into discrete values.

4. Predicting cost using C4.5 with incomplete data

C4.5 is an enhancement of the ID3 algorithm (Quinlan, 1986) that accounts for missing values, continuous feature values, pruning of decision trees, rule derivation, and so on. It builds decision trees top-down and prunes them. A tree is constructed by finding the highest information gain feature test to conduct at the root node of the tree. After the test is chosen, the cases are split according to the test, and the subproblems are solved recursively.

Suppose there are n_c classes in a given data set $\mathscr{X}$, and s_i $(i = 1, 2, \ldots, n_c)$ is the number of cases of in class C_i $(i = 1, 2, \ldots, n_c)$. The expected information needed to classify $\mathscr{X}$ is defined as

$$I(s_1, s_1, \ldots, s_{n_c}) = -\sum_{i=1}^{n_c} \frac{s_i}{n} \log_2 \frac{s_i}{n}, \tag{1}$$

where n is the total number of cases in data set $\mathscr{X}$. Note that a log function to the base 2 is used since the information is encoded in bits.

Let a feature f of $\mathscr{X}$ has v distinct values, $\{a_1, a_2, \ldots, a_v\}$. Feature f can be used to partition $\mathscr{X}$ into v subsets, $\{\mathscr{X}_1, \mathscr{X}_2, \ldots, \mathscr{X}_v\}$, where $\mathscr{X}_j$ $(j = 1, 2, \ldots, v)$ contains those cases in $\mathscr{X}$ that have value a_j $(j = 1, 2, \ldots, v)$ of f. Suppose $s_{i,j}$ is the number of cases of class C_i in a subset $\mathscr{X}_j$, the expected information needed to classify cases into subsets by feature f, also referred to as *Entropy*, is defined as

$$E(f) = \sum_{j=1}^{v} \frac{s_{1j} + s_{2j} + \cdots + s_{n_c j}}{n} I(s_{1j}, s_{2j}, \ldots, s_{n_c j}), \tag{2}$$

Where, for a given subset $\mathscr{X}_j$,

$$I(s_{1j}, s_{2j}, \ldots, s_{n_c j}) = -\sum_{i=1}^{n_c} \frac{s_{ij}}{s_j} \log_2 \frac{s_{ij}}{s_j}. \tag{3}$$

The information *Gain* of feature f is:

$$Gain(f) = I(s_1, s_2, \ldots, s_{n_c}) - E(f). \tag{4}$$

One limitation of *Gain* is that it tends to favor features with a large number of values that split the data into many small subsets. To compensate for this, Quinlan suggested using *Gain Ratio* instead of *Gain*. *Gain Ratio* is defined as

$$Gain\ Ratio(f) = \frac{Gain(f)}{SplitInfo(f)}, \tag{5}$$

where $SplitInfo(f)$ is the information due to the split of $\mathscr{X}$ on the basis of the values of f, it indicates the outcome of the test rather than the class to which the case belongs. Thus

$$SplitInfo(f) = I(|\mathscr{X}_1|, |\mathscr{X}_2|, \ldots, |\mathscr{X}_v|). \tag{6}$$

Sometimes the split is non-trivial and the split information $SplitInfo(f)$ will be small and this ratio will be unstable. To avoid this, the *Gain Ratio* criterion selects a test to maximize the ratio above, subject to constraint that the information gain must be greater than the average gain over all tests examined.

C4.5 deals with missing data with no assumption about the missing data mechanism. It uses a probabilistic approach to handle missing values, this approach consists of the following two steps.

The *first step* is the penalizing of *Gain Ratios*. Suppose in class C_i of subset $\mathscr{X}_j$, s_{ij}^m is the number of the cases whose feature f contains missing values. Let $\delta = \sum_{i=1}^{n_c} \sum_{j=1}^{v} \frac{s_{ij} - s_{ij}^m}{s_{ij}}$, the *Gain* is redefined as

$$Gain_m(f) = \delta \times (I(s_1, s_2, \ldots, s_{n_c}) - E(f)). \tag{7}$$

Suppose a subset of $\mathscr{X}_j$ is $\mathscr{X}_j^m$ which consists of cases whose values of f were missing, *SplitInfo* is redefined as

$$SplitInfo_m(f) = I(|\mathscr{X}_1 \ominus \mathscr{X}_1^m|, |\mathscr{X}_2 \ominus \mathscr{X}_2^m|, \ldots, |\mathscr{X}_v \ominus \mathscr{X}_v^m|) + I(|\mathscr{X}_1^m|, |\mathscr{X}_2^m|, \ldots, |\mathscr{X}_v^m|). \tag{8}$$

Therefore,

$$Gain\ Ratio_m(f) = \frac{Gain_m(f)}{SplitInfo_m(f)}. \tag{9}$$

The *second step* is the fractioning of cases. This step assigns a probability to each of the possible values $\{a_1, a_2, \ldots, a_v\}$ of f. These probabilities can be estimated based on the observed frequencies of the various values for feature f among the cases at a given node.

For example, suppose $f = \{a, b\}$ is a two-value feature, if a node contains seven known cases with $f = a$ and 3 with $f = b$, then the probability that $f(?) = a$[1] is 0.7, and the probability that $f(?) = b$ is 0.3. A fractional 0.7 of '?' is now distributed down the branch for $f = a$ and a fractional 0.3 of '?' down the other tree branch. These fractional cases are used for the purpose of computing information gain and can be further subdivided at subsequent branches of the tree if a second missing feature value must be tested. This same fractioning of cases can also be applied to classify new cases whose feature values were missing. In this case, the classification of the new cases is simply the most probable classification, computed by summing the weights of the case fragments classified in different ways at the leaf nodes of the tree.

Once the decision tree is obtained from the incomplete data, it is ready for us to predict software cost with it. However, when using the decision tree to predict software cost, we can not only assign a specific value for each missing data item, but also can designate a series of possible values and the corresponding probabilities for a categorical feature and an interval for a continuous feature. It seems to be one of the best amongst simple methods to treat missing values (Grzymala-Busse and Hu, 2000).

5. Experiments and results

5.1. Experimental method

5.1.1. General method

The objective of this study is to investigate the predictive performance of C4.5, which is a common machine learning technique that can tolerate missing values, after using the *k*-NN missing data imputation technique to see if it is better to tolerate missing data or to try to impute missing values and then apply the C4.5 algorithm in the context of software cost prediction.

For this purpose, first we preprocessed the six real world data sets (see Section 5.2 for details) and obtained the corresponding six complete data sets. Then, for each of the six complete data sets, the TwoStep clustering method (Austin and Belbin, 1982) with a log-likelihood distance measure was used to classify effort into clusters, and Yu and Liu's feature subset selection method FCBF (Yu and Liu, 2003) was used to choose key features for the software development effort prediction purpose, we obtained the six reduced data sets. After that, for each of the six reduced data sets, we systematically extracted 5 pairs of the training and test data sets (see Section 5.1.2 for details) and obtained a total of 30-pair training-test data sets. By simulating various missing data situations (see Section 5.1.3 for details) from the missingness mechanism, the missing data pattern, and the missing data percentage for the 30-pair training-test data sets, we obtained thousands of the incomplete training and test data sets. At the same time, by imputing these incomplete data sets with the *k*-NN method[2] we obtained the same amount of the imputed complete training and test data sets as the corresponding incomplete training and test data sets. Finally we ran C4.5 with three types of data sets: the complete, imputed complete, and incomplete data sets and obtained the corresponding prediction accuracies and summarized the results. Fig. 1 contains the details of the general research method.

5.1.2. Validation approach

Cross-validation is a method for estimating generalization error based on "resampling" (Shao and Tu, 1995). We used the 5-fold cross-validation strategy as the validation approach. In 5-fold cross-validation, the data set $\mathscr{D}$ is randomly split into 5 mutually exclusive subsets $\mathscr{D}_1, \mathscr{D}_2, \ldots, \mathscr{D}_5$ of equal size, and $\cup_{i=1}^{5} \mathscr{D}_i = \mathscr{D}$. The inducer is trained and tested five times. Each time $t \in \{1, 2, \ldots, 5\}$, it is trained on $\mathscr{D} \ominus \mathscr{D}_t$[3] which is referred to as a training set and tested on $\mathscr{D}_t$ which is referred to as a test set.

As filling in missing data is not the final objective which is using imputed complete data to do other things, for example, predicting software effort. Moreover, it has been proven that even a better estimate for each missing value not necessary leads to a better overall estimate for the parameters of interest (Selby and Porter, 1988). So using the results of a machine learning method to evaluate the missing data techniques is more practically useful and at least Tabachnick and Fidell (2001), Pickard et al. (2001) and Batista and Monard (2003) have used this method. Here, we used the prediction accuracy as a measure to evaluate the k-NN based imputation method and the C4.5 classifier with the complete, incomplete, and imputed complete data sets. The accuracy measure is defined as follows:

$$\text{Accuracy} = \frac{n_c}{n} \times 100\%,$$

where n_c is the number of cases whose class labels being correctly predicted and n is the total number of cases in a test set.

5.1.3. Missing data simulation approach

For the purpose of completely assessing the impacts of the missingness mechanism, the missing data pattern, and the missing data percentage on the performance of the missing data toleration technique of C4.5 and the *k*-NN missing data imputation technique in the context of software cost prediction, the simulation approach which was used for simulating various missing data situations was used.

When inducing missing values from both the complete training and test data sets, the missingness mechanism, the missing data pattern, the missing data percentage, and the number of features with missing data are the four parameters considered.

All the three missingness mechanisms MCAR, MAR, and NI were simulated.

1. The implementation of the MCAR mechanism is to induce missing values for the desired feature or features completely at random.
2. The implementation of the MAR mechanism is based on the size of the project. The bigger the project, the greater the probability of missing data. Specifically, first we order the cases according to project size. Then we divide the data set into 4 subsets with different percentages of missing data. The missing data percentage is proportion to the mean of project size of each subset. That is, for the *i*th subset, the missing data percentage is $\frac{M_i}{\sum_{j=1}^{4} M_j} \times p\% \times n$, where M_i is the mean of project size of the *i*th subset, *p*% is the given missing data percentage which is the amount of missing data we want to missing in a given data set under the given missingness mechanism and the given missing data pattern, and n is the number of data items.
3. The implementation of the NI mechanism is very similar to MAR. The only difference is the missing values were induced on the given particular features in question and only depend on the particular features themselves. Both the greater and smaller values (or the values with the greater and smaller frequency) have a higher likelihood of missingness.

Three missingness patterns were simulated, they are the 1-N pattern – one nominal feature with missing values, the 1-C pattern

[1] $f(?)$ means a missing value of feature f among cases at a given node.

[2] In the experiment, for the *k*-NN method, we used the two nearest cases ($k = 2$) since Kadoda et al. (2001) suggested this to perform consistently better than higher values of k for this particular problem domain.

[3] The notation $\mathscr{D} \ominus \mathscr{D}_t$ means set $\mathscr{D}$ minus set $\mathscr{D}_t$.

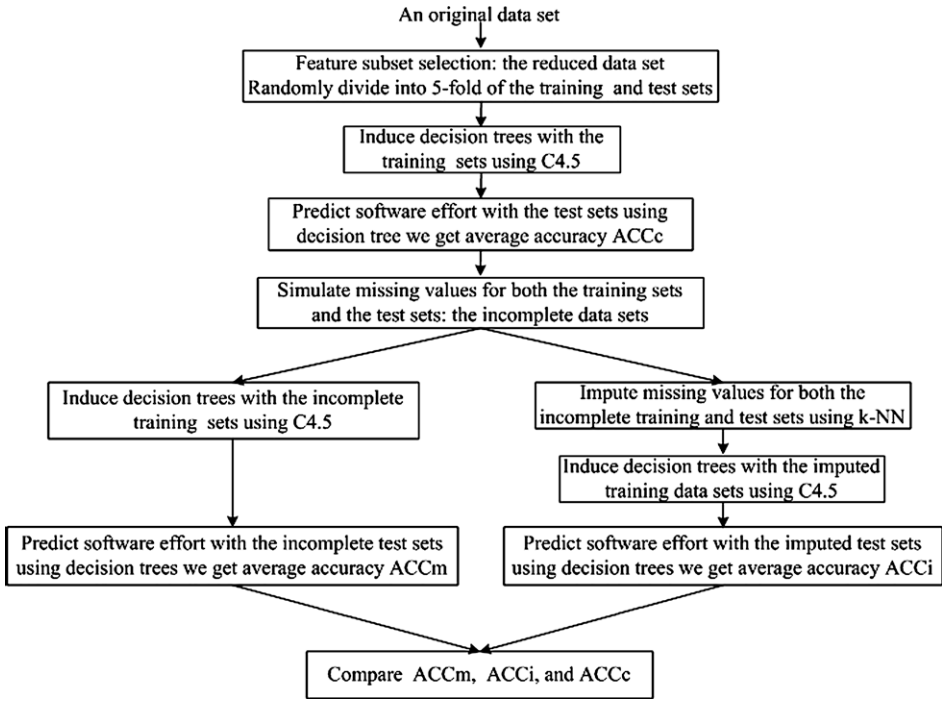

Fig. 1. The research method.

– one continuous feature with missing values, and the Arbitrary pattern – arbitrary features with missing values. For each of the three missing data patterns, the missing values were induced based on the given missingness mechanisms and the given missing data percentages. Moreover, for each of the three missingness patterns, missing values were induced for every feature. However, for the 1-N pattern and the 1-C pattern, every time only one feature contained missing values (but each feature has the chance to contain missing values) while all features contained missing values for the Arbitrary pattern.

Five missing data percentages, 10%, 20%, 30%, 40%, and 50% were simulated, although the data sets with more than 40% missing data percentage are not useful for detailed analysis (Strike et al., 2001).

5.2. Data sources

In this section we provide some background on the six data sets used in our study. These are chosen to represent varying sizes of data set that are commonly encountered in the project prediction domain. We selected two small sized, two medium sized and two large sized software project data sets for the analysis. These are chosen to represent a diverse range of software developers and a mixture of continuous, discrete and categorical features as well. The chosen enables us to generalize conclusions to other cases in the software project prediction domain.

5.2.1. Finnsh data set

Finnish is a large data set collected by the benchmarking organization STTF Ltd. This data is collected over a number of years from more than 30 different companies and more than 600 projects from insurance, banking, manufacturing, communications, retail and government sectors. The features are a mixture of continuous, discrete and categorical. However, there are a number of missing data values and also some features that would not be known at prediction time and so are not included in our analysis. Removing features with missing values or after-the-event data, leaves a subset of 42 features that are actually used in the study. The data set also exhibits significant multi-collinearity, in other words there are strong relationships between features as well as with effort.

5.2.2. ISBSG data set

ISBSG is an international organization, based on the national software metrics associations from several countries. ISBSG believes that they are representative of better software projects worldwide. Presently ISBSG Data Repository contains data on approximately 1238 projects with 55 features from insurance, government, banking, business services, manufacturing, communications, and utilities organizations of 20 countries. The projects cover a wide range of applications, development techniques and tools, implementation languages, and platforms. The features are a mixture of continuous, discrete and categorical as well.

5.2.3. Desharnais data set

Desharnais is a medium sized data set collected by a Canadia software house from projects distributed amongst 11 different organizations. This data set is publicly available, it contains 81 cases of which 77 are complete and combines both continuous and categorical features (8 continuous or discrete and 1 categorical). This data set exhibits properties that are representative of other data sets of software projects with respect to (non-)linearity and heteroscedasticity.

5.2.4. COCOMO81 data set

COCOMO81 is the data set that was used by Boehm (1984) to build the COCOMO model and also was used by Briand et al. (1992), Srinivasan and Fisher (1995) and Samson et al. (1997) to compare different effort prediction methods. It contains 63 projects and 40 features (23 continuous and 17 categorical).

5.2.5. BT data set

BT is a small data set derived from one division of a large telecommunication company. This is representative of many organiza-

tions that embark upon an internal data collection program to support their effort prediction activities. The data is relatively homogeneous, comprises 17 projects and 11 features (10 continuous and 1 categorical).

5.2.6. Albrecht data set

The Albrecht data set actually is the IBM DP Services data but was first used by Albrecht and Gaffney (1983) and was also used by Shepperd and Schofield (1997) to validate software size and effort estimation methods. The data comprises 23 projects and 8 features (7 continuous and 1 categorical).

5.3. Results

In this subsection, we present the experimental results for the six complete data sets and both the corresponding incomplete and imputed complete data sets with different missing data percentages and different missing patterns under all the three missingness mechanisms MCAR, MAR and NI for the C4.5 method. For each missing data percentage of each missing pattern under a specific missingness mechanism, we used the mean accuracy of five data replications. Then we summarized the results from the four aspects of the data set type, the missingness mechanism, the missing data pattern, and the missing data percentage (see Figs. 2–5 for details.)

Fig. 2 contains the accuracy of C4.5 with the six complete data sets and the corresponding incomplete and imputed complete data sets. From it we observe that, for each of the six data sets and its variations, the accuracy of C4.5 with the imputed complete data set is between with the complete data set and the incomplete data set; averagely, the accuracy has been improved by 6 percent. This means the *k*-NN imputation improves the performance of C4.5's missing data toleration technique, further improves the prediction accuracy of C4.5.

Fig. 3 contains the accuracy of C4.5 with the incomplete and imputed complete data sets under all the three missingness mechanisms MCAR, MAR, and NI. From it we observe that: (1) the accuracy of C4.5 with the incomplete data sets decreases as the missingness mechanism changes from MCAR through MAR to NI. This means that the missingness mechanism negatively affects the performance of C4.5's missing data toleration technique and further the accuracy of C4.5, and the NI missingness mechanism has a little stronger negative impact on prediction (or imputation) accuracy than the other two missingness mechanisms. (2) The accuracy of C4.5 with the imputed complete data sets also decreases as the missingness mechanism changes from MCAR through MAR to NI. This means that the missingness mechanism negatively affects the imputation accuracy of *k*-NN, and the NI missingness mechanism has a little stronger negative impact than the other two missingness mechanisms. (3) For each pair of the incomplete and imputed complete data sets under each missingness mechanism, the accuracy of C4.5 with the imputed complete data set is higher than that of with the incomplete data set. This

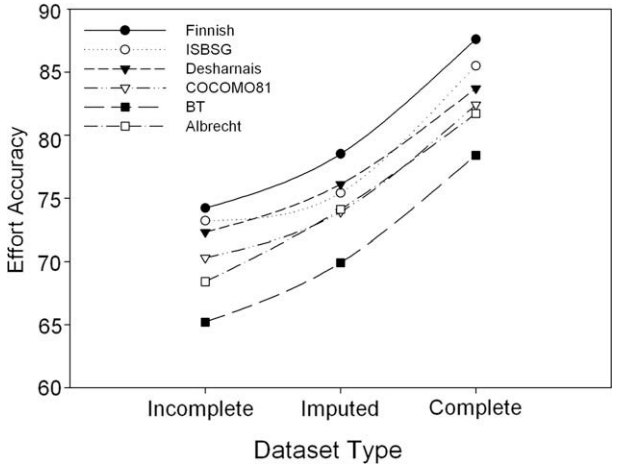

Fig. 2. Accuracy of C4.5 with the incomplete, imputed complete, and complete data sets.

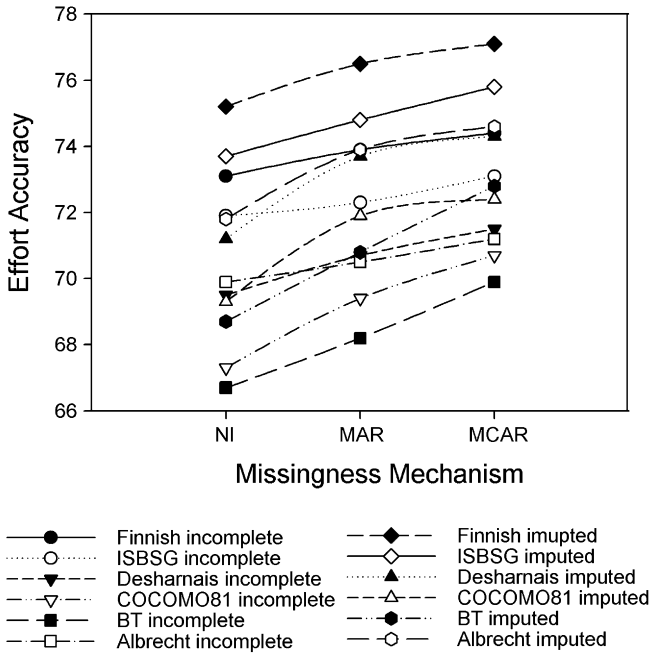

Fig. 3. Accuracy of C4.5 with the incomplete and imputed complete data sets under the three missingness mechanisms.

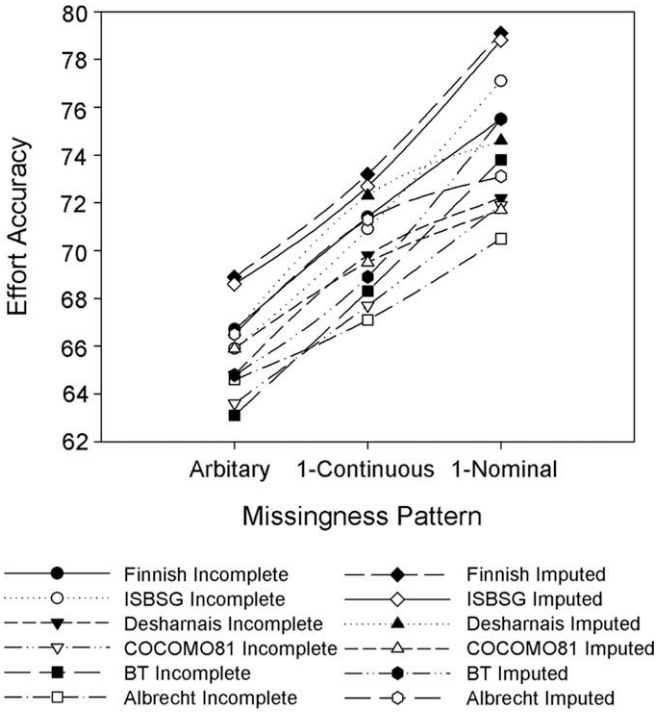

Fig. 4. Accuracy of C4.5 with the incomplete and imputed complete data sets under the three missing data patterns.

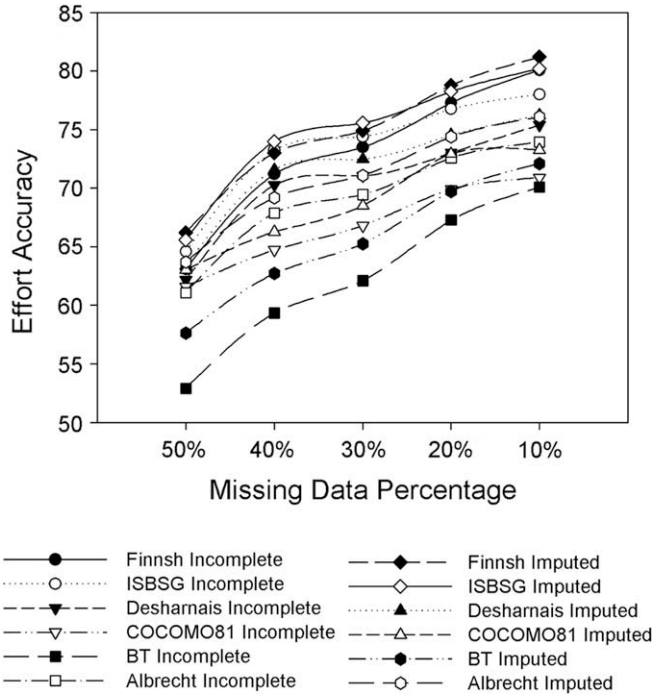

Fig. 5. Accuracy of C4.5 with the incomplete and imputed complete data sets under the five missing data percentages.

means that for all the three missingness mechanisms, the *k*-NN imputation improves the performance of C4.5's missing data toleration technique, further improves the prediction accuracy of C4.5.

Fig. 4 contains the accuracy of C4.5 with the incomplete and imputed complete data sets under all the three missing data patterns 1-N, 1-C and Arbitrary. From it we observe that: (1) the accuracy of C4.5 with the incomplete data sets decreases strongly as the missing data pattern changes from 1-N through 1-C to Arbitrary. This means the missing data pattern negatively affects the performance of C4.5's missing data toleration technique and further the accuracy of C4.5; and the impacts of these three patterns are quite different, the Arbitrary missing data pattern has a stronger negative impact on prediction (or imputation) accuracy than the other two missing data patterns. (2) The accuracy of C4.5 with the imputed complete data sets also decreases strongly as the missing data pattern changes from 1-N through 1-C to Arbitrary. This means the missing data pattern negatively affects the imputation accuracy of *k*-NN; and the impacts of these three patterns are quite different, the Arbitrary missing data pattern has a stronger negative impact on prediction accuracy than the other two missing data patterns. (3) For each pair of the incomplete and imputed complete data sets under each missingness pattern, the accuracy of C4.5 with the imputed complete data sets is higher than that of with the incomplete data sets. This means that for all the three missingness patterns, the *k*-NN imputation improves the performance of C4.5's missing data toleration technique, further improves prediction accuracy of C4.5, although the improvement for the 1-C and Arbitrary patterns are greater than for the 1-N pattern.

Fig. 5 contains the accuracy of C4.5 with the incomplete and imputed complete data sets under all the five missing data percentages 10%, 20%, 30%, 40%, and 50%. From it we observe that: (1) the accuracy of C4.5 with the incomplete data sets decreases as the missing data percentage increases. This means the missing data percentage negatively affects the performance of C4.5's missing data toleration technique and further the accuracy of C4.5; the bigger the missing data percentage, the less the prediction (or imputation) accuracy. More importantly, if the missing data percentage exceeds 40%, the accuracy will decrease greatly. (2) The accuracy of C4.5 with the imputed complete data sets also decreases as the missing data percentage increases. This means the missing data percentage negatively affects the imputation accuracy of *k*-NN; the bigger the missing data percentage, the less the imputation accuracy. More importantly, if the missing data percentage exceeds 40%, the imputation accuracy will decrease greatly. (3) For each pair of the incomplete and imputed complete data sets under each missing data percentage, the accuracy of C4.5 with the imputed complete data sets is higher than that of with the incomplete data sets. This means that for all the five missing data patterns, the *k*-NN imputation can always the performance of C4.5's missing data toleration technique and further the prediction accuracy of C4.5, but the improvement decreases as the missing data percentage increases.

Table 4
Mann–Whitney test of the prediction accuracy differences between the imputed complete data sets and the incomplete data sets regardless of whatever the missingness mechanism, the missing data pattern, and the missing data percentage are

Data set	*p*-Value
Finnsh	0.0042
ISBSG	0.0025
Desharnais	0.0157
COCOMO81	0.0003
BT	0.0235

For the purpose of more formally determining whether the accuracy improvements of the imputation are statistical significant compared with the missing data toleration technique of C4.5., we used a Mann–Whitney test, which is a non-parametric test that does not assume a Gaussian distribution, to compare the sample medians of the accuracies between the incomplete data sets and the imputed complete data sets. For all the tests, the null hypotheses are that there is no difference with ($\alpha = 0.05$).

In the Mann–Whitney test, for each of the six data sets, we suppose that the accuracy of C4.5 with the imputed complete data sets is greater than with the incomplete data sets regardless of whatever the missingness mechanism, the missing data pattern, and the missing data percentage are. Table 4 contains the corresponding test results. From it we observe that all the alternate hypotheses are accepted. This reveals that the improvements of imputation on prediction accuracy are statistical significant.

To summarize, from the results we observe that the *k*-NN imputation method can improve the performance of C4.5's missing data toleration technique and further the prediction accuracy of C4.5, regardless of whatever the missingness mechanism, the missing data pattern, and the missing data percentage are. More importantly, the improvements are statistical significant. At the same time, the missingness mechanism, the missing data pattern, and the missing data percentage negatively affect the prediction accuracy of C4.5 and imputation accuracy of *k*-NN; within each of these factors, the individual impact varies. On the other hand, for both C4.5 and *k*-NN, there was a big accuracy drop if the missing data percentage exceeds 40%.

6. Conclusions

In this study we have compared the impacts of the missing data toleration technique of C4.5 with the *k*-NN missing data imputation method on the prediction accuracy of C4.5 in the context of software cost prediction. This is further to help determine whether the *k*-NN imputation method can improve the performance of C4.5 when predicting with incomplete data.

First we induced the incomplete data sets from the six complete data sets for each combination of the missing data percentage, the missing data pattern and the missingness mechanism, and obtained thousands of incomplete training and test data sets. For the purpose of comparing the missing data toleration technique of C.45 with the *k*-NN imputation technique, we used the latter imputing these incomplete data sets and obtained the corresponding imputed complete training and test data sets. After that we assessed C4.5 with the incomplete data sets and the imputed completed data sets. We found that: (1) the *k*-NN imputation can improve the prediction accuracy of C4.5 and the improvements are statistical significant; (2) both C4.5 and *k*-NN can be affected by the missingness mechanism, the missing data pattern and the missing data percentage.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under Grants 60673124 and 90718024, the Hi-Tech Research & Development Program of China under Grant 2006AA01Z183, and the Program for New Century Excellent Talents in University under Grant NCET-07-0674. The authors would like to thank the anonymous reviewers for their helpful comments.

References

Albrecht, A., Gaffney, J., 1983. Software function, source lines of code, and development effort prediction. IEEE Transactions on Software Engineering 9 (6), 639–648.

Angelis, L., Stamelos, I., Morisio, M., 2001. Building a software cost estimation model based on categorical data. In: Proceedings of the Seventh International Software Metrics Symposium (METRICS 2001), pp. 4–15.

Austin, M., Belbin, L., 1982. A new approach to the species classification problem in floristic analysis. Australian Journal of Ecology 7, 75–89.

Baird, B., 1989. Managerial Decisions Under Uncertainty. John Wiley & Sons.

Batista, G.A., Monard, M.C., 2003. An analysis of four missing data treatment methods for learning. Applied Artificial Intelligence 17 (5–6), 519–533.

Boehm, B., 1981. Software Engineering Economics. Prentice-Hall, Englewood Cliffs, NJ.

Boehm, B.W., 1984. Software engineering economics. IEEE Transactions on Software Engineering 10 (1), 4–21.

Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J., 1984. Classification and Regression Trees. Wadsworth International Group, Belmont, CA.

Briand, L., 1993. Quantitative empirical modeling for managing software development: constraints, needs and solutions. In: Rombach, H.D., Basili, V., Selby, R. (Eds.), Experimental Software Engineering Issues: Critical Assessment and Future Directions. Springer-Verlag.

Briand, L.C., Wust, J., 2001. Modeling development cost in object-oriented systems using design properties. IEEE Transactions on Software Engineering 27 (11), 963–986.

Briand, L.C., Basili, V., Thomas, W., 1992. A pattern recognition approach for software engineering data analysis. IEEE Transactions on Software Engineering 18 (11), 931–942.

Briand, L.C., El Emam, K., Surmann, D. Wieczorek, I., Maxwell, K.D., 1999. An assessment and comparison of common software cost estimation modeling techniques. In: Proceedings of the 21st International Conference on Software Engineering, pp. 313–323.

Briand, L., El Emam, K., Wieczorek, I., 1999. Explaining the cost of european space and military projects. In: Proceedings of the International Conference on Software Engineering, pp. 303–312.

Briand, L., Langley, T., Wieczorek, I., 2000. A replicated assessment and comparison of common software cost modeling techniques. In: Proceedings of the 22nd International Conference on Software Engineering, pp. 377–386.

Brockmeier, L.L., Kromrey, J.D., Hines, C.V., 1998. Systematically missing data and multiple regression analysis: an empirical comparison of deletion and imputation techniques. Multiple Linear Regression Viewpoints 25, 20–39.

Cardie, C., 1993. Using decision trees to improve case-based learning. In: Proceedings of the Tenth International Conference on Machine Learning, Amherst, MA, pp. 25–32.

Cartwright, M., Shepperd, M., Song, Q., 2003. Dealing with missing software project data. In: Proceedings of the Ninth IEEE International Software Metrics Symposium. IEEE Computer Society, Sydney, Australia.

Cheung, R., Eisenstein, B., 1978. Feature selection via dynamic programming for text-independent speaker identification. IEEE Transactions on Acoustics, Speech and Signal Processing ASSP-26 (5), 397–403.

Clark, P., Niblett, T., 1989. The CN2 induction algorithm. Machine Learning 3 (4), 261–283.

Feelders, A.J., 1999. Handling missing data in trees: surrogate splits or statistical imputation? In: Proceedings of the Third European conference on principles and practice of knowledge discovery in data bases (PKDD99). Springer, pp. 329–334.

Finnie, G., Wittig, G., 1997. A comparison of software effort estimation techniques: using function points with neural networks, case-based reasoning and regression models. Journal of Systems and Software 39, 281–289.

Gama, J., Brazdil, P., 1999. Linear tree. Intelligent Data Analysis 3 (1), 1–29.

Gray, A., MacDonnell, D., 1997. A comparison of techniques for developing predictive models of software metrics. Information and Software Technology 39, 425–437.

Grzymala-Busse, J.W., Hu, M., 2000. A comparison of several approaches to missing attribute values in data mining. In: Proceedings of the Second International Conference on Rough Sets and Current Trends in Computing RSCTC'2000, October 16–19, 2000, Banff, Canada, pp. 340–347.

Holte, R.C., 1993. Very simple classification rules perform well on most commonly used datasets. Machine Learning 11 (1), 63–90.

Jain, A., Zongker, D., 1997. Feature selection: evaluation, application and small sample performance. IEEE Transactions on Pattern Analysis and Machine Intelligence 19 (2), 153–158.

Jeffery, R., Ruhe, M., Wieczorek, I., 2001. Using public domain metrics to estimate software development effort. Proceedings of the Seventh International Software Metrics Symposium (METRICS 2001), 16–27.

Jensen, R., 1983. An improved macrolevel software development resource estimation model. In: Proceedings of the Fifth ISPA Conference, pp. 88–92.

John, H.G., 1997. Enhancements to the Data Mining Process. PhD Thesis, Computer Science Department, School of Engineering, Stanford University, March 1997.

John, G., Kohavi, R., Pfleger, K., 1994. Irrelevant features and the subset selection problem. In: Cohen, W.W., Hirsh, H. (Eds.), Proceedings of the 11th International Conference on Machine Learning, San Fransisco, CA, pp. 121–129.

Jönsson, P., Wohlin, C., 2004. An Evaluation of *k*-Nearest Neighbour Imputation Using Likert Data. In: Proc. 10th IEEE International Software Metrics Symposium. IEEE Computer Society, Los Alamitos, California, USA, pp. 108–118.

Kadoda, G., Cartwright, M., Shepperd, M.J., 2001. Issues on the effective use of CBR technology for software project prediction. In: Proceedings of the Fourth International Conference on Case Based Reasoning, Vancouver.

Kemerer, C.F., 1987. An empirical validation of software cost estimation models. Communications of the ACM 30, 416–429.

Khotanzad, A., Kashyap, R., 1987. Feature selection for texture recognition based on image synthesis. IEEE Transactions on Systems, Man, and Cybernetics 17, 1087–1095.

Kibler, D., Aha, D.W., 1987. Learning representative exemplars of concepts: An initial case study. In: Proceedings of the Fourth International Workshop on Machine Learning. Morgan Kaufmann, Irvine, CA, pp. 24–30.

Kim, J.-O., Curry, J., 1977. The treatment of missing data in multivariate analysis. Sociological Methods and Research 6 (2), 215–240.

Kitchenham, B.A., 1992. Empirical studies of assumptions that underlie software cost estimation models. Information and Software Technology 34, 211–218.

Kohavi, R., John, G., 1997. Wrappers for feature subset selection. Artificial Intelligence Journal, 273–324.

Kubat, M., Flotzinger, D., Pfurtscheller, G., 1993. Discovering patterns in EEG-signals: comparative study of a few methods. In: Proceedings of the European Conference on Machine Learning ECML'93, Vienna, 3–7 April, pp. 366–371.

Little, R.J.A., 1988. A test of missing completely at random for multivariate data with missing values. Journal of The American Statistical Association 83 (404), 1198–1202.

Little, R.J.A., 1995. Modeling the drop-out mechanism in repeated-measures studies. Journal of the American Statistical Association 90, 1112–1121.

Little, R.J.A., Rubin, D.B., 1989. Analysis of social science data with missing values. Sociological Methods and Research 18, 292–326.

Little, R.J.A., Rubin, D.B., 2002. Statistical Analysis with Missing Data. John Wiley & Sons, New York.

Little, R.C., Roderick, J.A., Schenker, N., 1995. Missing data. In: Arminger, G., Clogg, C., Sobel, M. (Eds.), Handbook of Statistical Modeling for the Social and Behavioral Sciences. Plenum, New York, pp. 39–75.

Miranda, E., 2001. Improving subjective estimates using paired comparisons. IEEE Software 18 (1), 87–91.

Myrtveit, I., Stensrud, E., Olsson, U.H., 2001. Analyzing data sets with missing data: an empirical evaluation of imputation methods and likelihood-based methods. IEEE Transactions on Software Engineering 27 (11), 999–1013.

Pickard, L., Kitchenham, B., Linkman, S.J., 2001. Using simulated data sets to compare data analysis techniques used for software cost modeling. IEE Proceedings of the International Conference on Software 148 (6), 165–174.

Putnam, L.H., 1982. The real economics of software development. In: Goldberg, R., Lorin, H. (Eds.), The Economics of Information Processing. Wiley, New York.

Quinlan, J.R., 1986. Induction of decision trees. Machine Learning 1, 81–106.

Quinlan, J.R., 1993. C4.5 Programs for Machine Learning, Morgan Kaufmann, CA.

Robins, J.M., 1997. Non-response models for the analysis of non-monotone non-ignorable missing data. Statistics in Medicine 16, 21–38.

Rubin, D.B., 1976. Inference and missing data. Biometrika 63, 581–592.

Rubin, D.B., 1977. The design of a general and flexible system for handling nonresponse in sample surveys. Report prepared for the U.S. Social Security Administration.

Rubin, D.B., 1978. Multiple imputations in sample surveys – a phenomenological Bayesian approach to nonresponse. Proceedings of the Survey Research Methods Section of the American Statistical Association, 20–34.

Rubin, D.B., 1987. Multiple Imputation for Nonresponses in Surveys. John Wiley & Sons, New York.
Rubin, D.B., 1996. Multiple imputation after 18+ years (with discussion). Journal of the American Statistical Association 91, 473–489.
Samson, B., Ellison, D., Dugard, P., 1997. Software coast estimation using an Albus Perceptron (CMAC). Information and Software Technology 39 (1/2).
Selby, R.W., Porter, A.A., 1988. Learning from examples: generation and evaluation of decision trees for software resource analysis. IEEE Transactions on Software Engineering 14 (12), 1743–1757.
Shao, J., Tu, D., 1995. The Jackknife and Bootstrap. Springer-Verlag, New York.
Shepperd, M., Cartwright, M., 2001. Predicting with sparse data. IEEE Transactions on Software Engineering 27 (11), 987–998.
Shepperd, M., Schofield, C., 1997. Estimating software project effort using analogies. IEEE Transactions on Software Engineering 23 (12), 736–743.
Siedelecki, W., Skalansky, J., 1988. On automatic feature selection. International Journal of Pattern Recognition and Artificial Intelligence 2 (2), 197–220.
Song, Q., Shepperd, M., 2007. A new method for imputing small software project data sets. Journal of Systems and Software 80 (1), 51–62.
Song, Q., Shepperd, M., Cartwright, M., 2005. A short note on safest default missingness mechanism assumptions. Empirical Software Engineering: An International Journal 10 (2), 235–243.
Srinivasan, K., Fisher, D., 1995. Machine learning approaches to estimating software development effort. IEEE Transactions on Software Engineering 21 (2), 126–137.
Strike, K., El Emam, K., Madhavji, N., 2001. Software cost estimation with incomplete data. IEEE Transactions on Software Engineering 27 (10), 890–908.
Tabachnick, B.G., Fidell, L.S., 2001. Using Multivariate Statistics, 4th ed. Allyn & Bacon, Needham Heights, MA.
Tay, F., Shen, L., 2002. A modified chi-squared algorithm for discretization. IEEE Transactions on Knowledge and Data Engineering 14 (3), 666–670.
Tian, J., Porter, A., Zelkowitz, M. V., 1992. An improved classification tree analysis of high cost modules based upon an axiomatic definition of complexity. In: Proceedings of the Third International Symposium on Software Reliability Engineering, pp. 164–172.
Walkerden, F., Jeffery, R., 1997. Software cost estimation: a review of models, process, and practice. Advances in Computers 44, 59–125.
Walston, C., Felix, C., 1977. A method of programming measurement and estimation. IBM Systems Journal 1, 54–73.
Yu, L., Liu, H., 2003. Feature selection for high-dimensional data: a fast correlation-based filter solution. In: Proceedings of the Twentieth International Conference on Machine Leaning (ICML-03), Washington, DC, August 21–24, pp. 856–863.

Qinbao Song received a PhD in computer science from the Xi'an Jiaotong University, Xi'an, China in 2001. He is a professor of software technology in the department of computer science and technology at Xi'an Jiaotong University, Xi'an, China. He has published more than 70 referred papers in the area of data mining, machine learning, and software engineering. He is a Board Member of the Open Software Engineering Journal. His research interests include intelligent computing, machine learning for software engineering, and trustworthy software.

Martin Shepperd received a PhD in computer science from the Open University in 1991 for his work in measurement theory and its application to software engineering. He is professor of software technology at Brunel University, London, UK and director of the Brunel Software Engineering Research Centre (B-SERC). He has published more than 90 refereed papers and three books in the area of empirical software engineering, machine learning and statistics. He is editor-in-chief of the journal Information & Software Technology and was Associate Editor of IEEE Transactions on Software Engineering. He has also previously worked for a number of years as a software developer for a major bank.

Xiangru Chen received a BSc in computer science from the Xi'an University of Technology, Xi'an, China in 2006. He is a research student of computer science at Xi'an Jiaotong University, Xi'an, China.

Jun Liu received a MSc in computer science from the Xi'an Jiaotong University, Xi'an, China in 2008. She is a senior lecturer of computer science at Shaanxi Electric Power Training Center for the Staff Members, Xi'an, China.