Using Bayesian regression and EM algorithm with missing handling for software effort prediction

Wen Zhang\textsuperscript{a,}\textsuperscript{*}, Ye Yang\textsuperscript{b}, Qing Wang\textsuperscript{c}

\textsuperscript{a}School of Economics and Management, Beijing University of Chemical Technology, Beijing 100029, PR China
\textsuperscript{b}School of Systems and Enterprises, Stevens Institute of Technology, Hoboken, NJ 07030, USA
\textsuperscript{c}Laboratory for Internet Software Technologies, Institute of Software, Chinese Academy of Sciences, Beijing 100190, PR China

\textbf{Article info}

Article history:
Received 25 April 2014
Received in revised form 5 September 2014
Accepted 11 October 2014
Available online 18 October 2014

Keywords:
Bayesian regression
EM algorithm
Missing imputation
Software effort prediction

\textbf{Abstract}

Context: Although independent imputation techniques are comprehensively studied in software effort prediction, there are few studies on embedded methods in dealing with missing data in software effort prediction.

Objective: We propose BREM (Bayesian Regression and Expectation Maximization) algorithm for software effort prediction and two embedded strategies to handle missing data.

Method: The MDT (Missing Data Toleration) strategy ignores the missing data when using BREM for software effort prediction and the MDI (Missing Data Imputation) strategy uses observed data to impute missing data in an iterative manner while elaborating the predictive model.

Results: Experiments on the ISBSG and CSBSG datasets demonstrate that when there are no missing values in historical dataset, BREM outperforms LR (Linear Regression), BR (Bayesian Regression), SVR (Support Vector Regression) and M5\textsuperscript{0} regression tree in software effort prediction on the condition that the test set is not greater than 30\% of the whole historical dataset for ISBSG dataset and 25\% of the whole historical dataset for CSBSG dataset. When there are missing values in historical datasets, BREM with the MDT and MDI strategies significantly outperforms those independent imputation techniques, including MI, BMI, CMI, MINI and M5\textsuperscript{0}. Moreover, the MDI strategy provides BREM with more accurate imputation for the missing values than those given by the independent missing imputation techniques on the condition that the level of missing data in training set is not larger than 10\% for both ISBSG and CSBSG datasets.

Conclusion: The experimental results suggest that BREM is promising in software effort prediction. When there are missing values, the MDI strategy is preferred to be embedded with BREM.

\textcopyright 2014 Elsevier B.V. All rights reserved.

1. Introduction

Software effort prediction refers to estimate the human effort needed to develop a software artifact \cite{13, 22}. Overestimate of software effort lead to noncompetitive contract bids and over allocation of development resources and personnel. Whereas, underestimate of effort lead to tight schedule of development and compromise full functional development and thorough test of software resulting in faults in the system after delivery \cite{38}. Thus, the importance of software effort estimation \cite{3} motivates researchers using knowledge on software attributes and effort stored in historical datasets to develop predictive models, either by statistical methods such as linear regression \cite{6} and multiple regressions \cite{33}, or by machine learning methods such as active learning \cite{16} and transfer learning \cite{18}, to predict effort of future software projects.

However, one difficulty confronted with researchers in constructing the effort prediction model lies in that, more often than not, historical datasets contain a large amount of missing data caused by measurement noise or data corruption \cite{23}. For instance, the ISBSG (International Software Benchmarking Standard Group) dataset contains more than 40\% missing data in some attributes \cite{24}. The CSBSG (Chinese Software Benchmarking Standard Group) dataset contains more than 30\% missing data in some attributes and projects even after we remove those projects and attributes with more than 2/3 missing values \cite{41}. The common practice of ignoring projects with missing values (i.e. listwise deletion, LD) lead to a biased predictive model due to small size of most
historical datasets [24,39,36]. For this reason, the study of handling missing data in software effort prediction is becoming an attractive topic.

Generally, there are two problems involved in software effort prediction with missing data. The first problem is to handle missing data in dataset and, the second problem is to construct a feasible predictive model to characterize relationship between project attributes and needed effort, which is usually denoted by number of man month.

In the former aspect, missing data can be handled using either embedded method or independent method [11]. On embedded method, C4.5 [28] employs missing data toleration and ID3 [26] uses case deletion in building decision tree. On preprocessing method, LD and some imputation methods like BMI (Bayesian Multiple Imputation) [15] and MINI (class mean imputation based on k-NN hot deck imputation) [36] are usually used to fill in incomplete observations.

On the one hand, the independent methods are extensively studied in software effort prediction such as LD [24], BMI [15], CMI (Class Mean Imputation) and MINI [36]. A general conclusion that can be drawn from these studies is that BMI and MINI outperform other techniques in handling missing data in software effort prediction. However, on the other hand, there are few studies on embedded methods in dealing with missing data in software effort prediction.

In our prior work [42], we formulate software effort prediction as a classification problem in machine learning and propose MDT and MDI strategies to handle missing values. We discretize all the continuous attributes as well as the software effort to make software effort can be predicted using classical naïve Bayes as used in text classification. However, software effort prediction is different from typical classification in that it involves not only numeric attributes but also categorical attributes to characterize software projects (see [41]). Nevertheless, software effort classification is of less practical usefulness than effort regression. These are the reasons we extend our prior work in Zhang et al. [42] to use Bayesian regression for software effort prediction and consider both numeric and categorical attributes of software projects.

Bayesian regression has been proved in machine learning area that it has better performances than LR and CART in predicting numeric output with respect to the absolute error [9]. Due to the similarity of the expected absolute error (see Eq. (9)) and the magnitude of relative error (see Eq. (2)) in mathematical formulation, we conceive that Bayesian regression might be more appropriate than other methods in predicting software effort. Motivated by this, we propose to use Bayesian regression for software effort prediction and two embedded strategies are proposed to handle missing data.

The remainder of this paper is organized as follows. Section 2 introduces some preliminaries to understand the paper. Section 3 describes BREM for software effort prediction. Section 4 proposes the MDT and MDI strategies for handling missing data when using BREM for software effort prediction. Section 5 conducts experiments to examine BREM for effort prediction as well as the MDT and MDI strategies in handling missing data. Section 6 concludes the paper.

2. Preliminaries

2.1. Related work

Many researches have been conducted on handling missing data in software effort prediction. Song and Shepperd [36] propose a new imitation method called MINI to handle missing data in effort prediction (described in Section 2.2). New extensions and experiments of this method are also proposed in their recent work [37]. In essence, this method is a little similar to the MDI strategy proposed in this paper where missing values are imputed by expected values given project effort.

Strike et al. [39] evaluate ten usually adopted techniques for handling missing data: listwise deletion (LD), mean imputation (MI) and eight different types of hot-deck imputation techniques. Their results indicated that all the techniques performed well with small bias and high precision. The precision of LD is worse than the other techniques, especially as the percentage of missing data increases. The hot-deck imputation techniques that use Euclidean distance and z-score standardization to measure project similarities produce best performance. Since the hot-deck imputation technique uses the values of a project (project I), that has no missing data and is most similar with the project (project II) having missing data, to impute the missing values of the project (project II), we deduce that performance of prediction model would be improved if we use values from similar projects to impute missing values. However, according to Schafer and Graham [32], hot-deck imputation technique has no parametric model and distorts correlations and other measures of association of project attributes. For this reason, we propose to use expectation of attributes given effort to impute missing values and thus, missing data would have different values when they are assumed with different effort. Myrtreit et al. [24] conduct an empirical evaluation of imputation methods and likelihood-based methods including LD, MI, similar response pattern imputation (SRPI) and full information maximum likelihood (FIML). Their SRPI can be regarded as the same as hot-deck imputation based on k nearest neighborhood. Their experiments are carried out using the ISBSG dataset and it is reported that only FIML was appropriate when the missing data are not MAR. LD, MI and SRPI make predictive model biased unless the data are MACR. FIML assumes that the data comes from a multivariate normal distribution and maximizes the likelihood of the predictive model given the observed data. This is also the basic principle of EM algorithm when used for handling incomplete data. The superiority of FIML over other imputation methods also motivates us to use EM algorithm as the basis for handling missing data. Both FIML and EM have the same mechanism supported by maximization likelihood theory [1,20,2], and when the data are not MACR, they will not produce a biased predictive model. We have not used FIML for comparison in the experiments because on the one hand, it does not provide missing imputation and on the other hand, we have already used it in BMI for parameter initialization.

Twala et al. [40] compare seven missing handling techniques (LD, SPRI, FIML, MI, etc.) to handle incomplete data in software engineering. They report that among all the techniques, BMI technique achieve the best results. The ensemble of the two missing handling methods as BMI and kNNSI produce a significant improvement in effort prediction accuracy.

Khoshgoftaar and Hulse [15] also report that BMI technique outperforms other techniques using NASA data. They point out that the amount of missing data and the missingness mechanism are important factors in missing imputation process. All of their investigated imputation techniques (BMI, kNNSI, and MI) produce better performances when the missing data is under MACR than those are under MAR and MNAR. Only when the missing data is under MACR and at high missing level, it is necessary to use a complicated technique (such as kNNSI). Otherwise, BMI may be more preferable. We also compare the proposed MDI and MDT strategies embedded in BREM with BMI technique in this paper.

In our another prior work [41], we consider two types of mechanisms that may come up with missing data in effort datasets as absent features (structural missingness) and unobserved values (unstructured missingness). We employ two different techniques as max-margin regression and missing imputation to deal with
these two types of missingness. Experimental results on ISBSG and CSBSG datasets demonstrate that the treatment that regards miss-
ingness in software effort datasets as unobserved values produced more desirable performances than those treatments that regard missingness as absent features. These results inspire us to develop more elaborate missing handling techniques than preprocessing techniques in software effort prediction. That is, how to construct a predictive model that can on the one hand make accurate effort prediction and on the other hand handle missing data appropriately.

2.2. Problem formulation

Typically, we can divide a dataset Dcom into two parts: the one is observed without missing data, i.e. Dobs, and the other one is unobserved with missing data Dmis. That is, Dcom = (Dobs, Dmis). We can categorize the missingness mechanism of missing data into three categories: missing at complete random (MAR), missing at random (MAR) and missing not at random (MNAR) [29]. MAR refers to that distribution of missing data is dependent on neither of Dobs nor Dmis. MAR refers to that distribution of missing data is dependent on Dmis. That is, missing data appears dependent on values of corresponding observed data. MNAR refers to that distribution of missing data is dependent on Dmis and that, missing data appears dependent on its real values. For instance, for some reasons, a software engineer purposely removes low effort of his or her artifacts and keeps them missing.

The problem of handling missing data in effort prediction can be formulated as follows. For simplicity, we regard effort prediction as a task to predict a real numeric effort of a new incoming software project and, it is usually implemented by a regression method. We assume that a dataset Dcom containing m historical projects as Dcom = (D1,..., Dm)T, where Dl (1 ≤ i ≤ m) is a historical project and Dl = (x1l,..., xnjl,..., xNjl)T is represented by n attributes xjl (1 ≤ i ≤ n). Furthermore, we assume that h0l denotes the real effort of project Dl. For each xjl which is the value of attribute xjl (1 ≤ j ≤ n) on Dl, it can be observed or missing. Thus, the problem is transferred to handle the missing xjl in the historical dataset Dcom when using Dcom to construct predictive model M to predict h0l as accurate as possible.

In order to evaluate performance of the effort prediction method M, we usually divide the whole historical dataset Dcom into two sets. The one is training set used to training the predictive model for effort prediction and, the other one is test set used to evaluate the performance of the trained model on predicting software effort. Thus, we can rewrite Dcom as Dcom = (Dcom|Dtest) = (D1,..., Dm|Dtest)T, where m and m’ are the predefined numbers of projects in the training and test sets, respectively. For instance, in 10-fold cross validation, m’ is predefined as 0.1(m + m’). The difference of a training project Djl (1 ≤ i ≤ m) and a test project Djl (1 ≤ i ≤ m’) lies in that, the effort h0l is known for Djl but, the effort hjl of test project Djl is assumed as unknown. By combining missing handling and machine learning, an effort prediction model denoted by M is produced using the training set. The performance of effort prediction model M is usually evaluated by PRED(25) which is the percentage of predictions falling within 25% of the actual values as shown in Eq. (1) [16].

\[
PRED(25) = \frac{100}{m'} \sum_{i=1}^{m'} \begin{cases} 1, \quad \text{if } MRE_i \leq 0.25 \\ 0, \quad \text{otherwise.}\end{cases}
\]  

Here, MREi defined in Eq. (2) measures the difference between the predicted effort hjl and the real effort h0l of test project Djl. We can see that Eq. (2) is essentially a variant of the absolute error in constructing Bayesian regression described in Eq. (9). Considering the widely recognized performance measure as MRE and PRED(25) in software effort prediction, we claim that the adoption of Bayesian regression for effort prediction is reasonable.

\[
MRE_i = \frac{|h_{jl} - h_{0l}|}{h_{0l}}
\]  

2.3. Imputation techniques

The independent missing imputation techniques we discuss in this paper include MI (Mean Imputation), CMI (Class Mean Imputation), MINI (class mean imputation based on the k-NN hot deck imputation) and BMI (Bayesian Multiple Imputation). Although kNNSI (k-nearest neighbor single imputation) is also a usually adopted technique to impute unobserved value, we will not cover it in the paper because many studies have already proved that MINI and BMI outperforms KNNSI in missing imputation [24,15]. Nevertheless, we regard that MINI is actually an improved variant of kNNSI based on feature selection to measure project similarity.

MI imputes each missing value with the mean of observed values. Assuming we have a historical project Dl and its value on attribute xjl, i.e. xjl, is missing. Then, MI will use xjl = \frac{1}{|Dobs|} \sum_{j \in Dobs} xjl to impute xjl. Dobs is the set of projects whose value on attribute xjl is observed. This method is very simple and easy to implement but the variance of imputed values will be underestimated because all the missing values of an attribute xjl will be imputed using the same value xjl.

CMI imputes a missing value of a historical project Dl with the mean of observed values of projects in same class as the project Dl. For simplicity, let Dl denote the set of projects whose classes are the same as that of Dl. Assuming that xjl is unobserved, then CMI will impute xjl using xjl = \frac{1}{|Dobs|} \sum_{j \in Dobs} xjl to impute xjl. Dobs is the set of projects whose values on attribute xjl are observed. In order to overcome the problem of underestimation of variance of missing values in MI, CMI classify the historical projects into different classes firstly and then use the mean of the observed values under the attribute xjl and of projects with same effort class to impute the missing value xjl.

MINI regards the improvement of variance by CMI is not enough and should be improved further. Actually, there are two main procedures in MINI technique. The first procedure is key feature selection using information gain. The basic idea is to rank attributes according to the information they contain in classifying historical projects, with the goal of removing the redundant attributes in the dataset. The second procedure is to use k-nearest neighbor (KNN) method to select top similar projects of the project Dl to impute the missing value of Xjl. In Song and Shepperd [36], MINI imputes a missing value of a project using the average of values from its 2 most similar projects under the corresponding attributes. For space limitation, one of interest can refer to Section 3.2 of Song and Shepperd [36] for more details of MINI algorithm.

BMI is proposed by Rubin [30] and Schafer [31]. The basic idea of BMI is to generate multiple imputations for a missing value from the predictive posterior distribution of the missing data given the observed data. Then, the missing value is imputed by the average of these synthetic values. For a missing value xjl in the dataset, we will create r imputed values xjl(1),..., xjl(r) by sampling the donotions from the posterior predictive distribution of the missing data P(Dmis|P(obs)), Dmis, and P(obs) will be explained in Section 4. Usually, the data augmentation (DA) technique, that includes the I-(or Imputation) step and the P-(or Posterior) step, is used to generate the r datasets as follows. Here, \( \Theta \) represents the set of unknown population parameters, and \( \theta \) denotes the number of iteration. We assume that the effort dataset conforms to multivariate normal distribution and thus \( \Theta \) denotes the mean and covariance matrix.
(I) Suppose that the current estimate for parameters \( \Theta \) are given by \( \Theta \) (0). Sample an estimate of the missing values \( D_{mis}^{(0)} \) from the distribution \( P(\Theta(0)|D_{mis}, D_{obs}) \). 
(P) Using the estimate of the missing values, compute \( \Theta(1) \) from \( P(\Theta(1)|D_{mis}, D_{obs}) \).

The I-step is to sample the missing values using the estimate for parameter \( \Theta \) and the observed data set \( D_{obs} \). The P-step makes use of the observed data \( D_{obs} \) and assumed values of \( D_{mis} \) to produce a complete data posterior \( P(\Theta|D_{mis}, D_{obs}) \). A sequence of datasets \( D_{mis}^{(1)} \ldots D_{mis}^{(m)} \) is sampled from \( P(D_{mis}|D_{obs}, \Theta^{(0)}) \ldots P(D_{mis}|D_{obs}, \Theta^{(m-1)}) \). Note that DA is an iterative procedure where the current sampling from \( P(D_{mis}|D_{obs}, \Theta^{(r)}) \) relies on the previous sampling \( P(D_{mis}|D_{obs}, \Theta^{(r-1)}) \). \( D_{mis}^{(r)} \) and \( D_{mis}^{(r)} \) are dependent on each other if \( i \) and \( j \) are close. For this reason, Bayesian proper is defined by Schafer [31] which attempts to produce independent multiple imputations of missing data \( D_{mis} \).

Sequential and parallel chains are often used to produce Bayesian proper multiple imputations with data augmentation. In this paper, parallel chains, that require the execution of multiple runs of BMI, are used to generate Bayesian proper. The sampled dataset at the final iteration is the one that will be averaged for imputation. When \( r \) is large enough, then \( D_{mis}^{(r)} \) is a random draw from the distribution \( P(D_{mis}|D_{obs}, \Theta^{(r-1)}) \). The parameters \( \Theta^{(0)} \) can be initialized using the Expectation Maximization (EM) algorithm by direct computation with observed values after convergence or the FIML (Full Information Maximum Likelihood) by maximum likelihood estimation with incomplete data.

### 3. Bayesian regression with EM algorithm

#### 3.1. Using Bayesian regression for effort prediction

Naïve Bayes [8] is a well-known probabilistic classifier in data mining. For computation simplicity, it assumes that the attributes of projects are independent with each other. Despite this assumption violates the realities in most applications, naïve Bayes performs more accurate than other sophisticated methods in many practical applications such as text classification [19] and kernel density estimation [14]. In software effort prediction, we also assume that the attributes (such as project size, development type and business area) of projects independent of each other.

We describe the historical effort dataset conceptually in Table 1 and follow the definition in Section 1 that \( X_j \) (1 ≤ \( j \) ≤ \( n \)) denotes an attribute of project \( D_i \) (1 ≤ \( i \) ≤ \( m \)) and \( h_i \) denotes the real effort of \( D_i \). All the numeric attributes and the effort of projects are normalized to [0,1] by their range.

The likelihood \( P(X_j=x|H=h) \) of \( X_j=x \) with respect to effort \( h \) is estimated using kernel density estimation [35]. If \( X_j \) is a numeric attribute, we estimate the likelihood \( P(X_j=x|H=h) \) using Eqs. (3)–(5).

\[
P(X_j=x|H=h) = \frac{1}{mb_{X_j}h} \sum_{i=1}^{m} K(x-x_i/b_{X_j}) K(h-h_i/b_h) \tag{3}
\]

\[
P(H=h) = \frac{1}{mb_{h}} \sum_{i=1}^{m} K(h-h_i/b_h) \tag{4}
\]

\[
P(X_j=x|H=h) = \frac{P(X_j=x|H=h)}{P(H=h)} \tag{5}
\]

Here, \( b_{X_j} \) and \( b_h \) are numeric bandwidths [35] for attribute \( X_j \) and effort \( h \), respectively. To simplify, we set \( b_{X_j} = 1.06\sigma_{X_j}m^{-1/5} \) and \( b_h = 1.06\sigma_h m^{-1/5} \) suggested by Silverman [35] where \( \sigma_{X_j} \) and \( \sigma_h \) are the standard deviations of observed values of the attribute \( X_j \) and the real software effort under \( h \), respectively. The kernel function \( K(.) \) is given by the Gaussian kernel as \( K(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right) \) suggested by Frank et al. [9] due to its superior mathematical properties.

If \( X_j \) is a categorical attribute, then the likelihood \( P(X_j=x|H=h) \) of \( X_j=x \) with respect to effort \( h \) is estimated using Eqs. (6)–(8).

\[
P(X_j=x) = \frac{t(x_i=x)}{m} \tag{6}
\]

\[
P(H=h|X_j=x) = \frac{1}{t(x_i=x) b_{X_j} b_h} \sum_{x_i=x} K_h(h-h_i/b_h) \tag{7}
\]

\[
P(X_j=x|H=h) = \frac{P(X_j=x|H=h)P(H=h|X_j=x)}{\sum_{x_i=x} P(x_i=x)P(H=h|x_i=x)} \tag{8}
\]

Here, \( t(x_i=x) \) is the number of projects \( D_i \) whose value under the attribute \( X_j \) is equal to \( x_i \), i.e. \( x_i = x \), \( b_{X_j} b_h \) is the bandwidth for effort of projects whose values under the attribute \( X_j \) are equal to \( x_i \), i.e. \( b_{X_j} b_h = 1.06\sigma_{X_j} \sigma_h^{-1} \). We adopt the same method in producing the bandwidths as used for numeric attributes. Because each software effort is a numeric value, we adopt the Eq. (4) directly to estimate the prior probability \( P(H=h) \).

Naïve Bayes is typically used in classification. However, software effort estimation is to predict a real number for each software project. For this reason, we adopt Bayesian regression [9] to predict software effort. We estimate the effort by rule of minimizing the expected absolute error. That is, for a project \( D_i \) with unknown effort \( h_i \), we need to estimate a real effort \( \hat{h}_i \) that minimizes the expected absolute error described in Eq. (9).

\[
E[|\hat{h}_i - h_i|] = \int P(H=h|D_i) |\hat{h}_i - h_i| dh_i \tag{9}
\]

We can easily show that the minimum of Eq. (9) is achieved when the condition \( \int_{\hat{h}_i}^{h_i} P(H=h|D_i) dh_i = 0.5 \) meets. This condition also means that the optimum \( \hat{h}_i \) is the median effort given \( D_i \). Thus, it inspires us to find the \( \hat{h}_i \) by Eq. (10), i.e. the effort predictive model M.

\[
\sum_{h_i \in G} P(H=h|D_i) > 0.5 \tag{10}
\]

Following Frank et al. [9], G is predefined as a series of grid points of real effort \( H \), with \( G = \{h_{min} - 2t, h_{min} - t, h_{min}, h_{max}, h_{max} - t, \ldots, h_{max} - t, h_{max}\} \). \( h_{max} \) and \( h_{min} \) are the maximum and minimum effort in all the training software projects, respectively. We set \( t = (h_{max} - h_{min})(d-1) \) and \( d = 100 \) in this paper. The joint probability \( P(H=h|D_i) \) is estimated using Eq. (11).

\[
P(H=h,D_i) = P(H=h)P(D_i|H=h) = P(H=h) \prod_{j=1}^{n} P(X_j=x_j|H=h) \tag{11}
\]

| Table 1 The historical effort dataset of software projects. |
|-----------------|-----------------|-----------------|-----------------|-----------------|
| \( X_1 \) | \( X_2 \) | \( X_3 \) | \( X_4 \) | \( H \) |
| \( D_1 \) | \( x_{11} \) | \( x_{12} \) | \( x_{13} \) | \( x_{14} \) | \( h_1 \) |
| \( D_2 \) | \( x_{21} \) | \( x_{22} \) | \( x_{23} \) | \( x_{24} \) | \( h_2 \) |
| \( \ldots \) | \( \ldots \) | \( \ldots \) | \( \ldots \) | \( \ldots \) | \( \ldots \) |
| \( D_m \) | \( x_{m1} \) | \( x_{m2} \) | \( x_{m3} \) | \( x_{m4} \) | \( h_m \) |
Here, \( P(H = h) \) is estimated using Eq. (4). If \( X_j \) is a numeric attribute in \( D_i \), then we use Eq. (5) to estimate \( P(X_j = x_j | H = h) \). If \( X_j \) is a categorical attribute in \( D_i \), then we use Eq. (8) to estimate \( P(X_j = x_j | H = h) \). In this way, the real effort of project \( D_i \) is estimated as \( h_{D_i} \).

3.2. Integrating Bayesian regression with EM

EM is an iterative algorithm using maximum likelihood or maximizing a posterior estimation in problems with incomplete data [25,7]. In effort prediction, projects \( D_i (1 \leq i \leq m) \) with unknown effort in the test set are regarded as incomplete data. In Section 3.1, Bayesian regression uses projects \( D_i (1 \leq i \leq m) \) with known effort to train effort prediction model \( M \). However, although we minimize the absolute error of prediction in Eq. (9), the prediction performance in Eq. (10) will still suffer because variance in the estimates of the predictive model such as those parameters in Eqs. (11)–(14) will be high due to limited number of observed projects for software effort estimation [25].

The trick to alleviate this suffering is to include the test project \( D_i \) in training predictive model. That is, to integrate Bayesian regression with EM algorithm. The motivation is quite straightforward, to reduce the variance of estimates of the predictive model.

The difference of a training project \( D_i \) and a test project \( D_{test} \) is that \( h_i \) is known for project \( D_i \) but \( h_i \) is assumed as unknown for \( D_{test} \). However, if we use Eq. (10) to make a prediction for \( D_i \), then \( h_{D_i} \) could also be used in Eqs. (3), (4), (5), (7) and (8) to update \( P(X_j = x | H = h) \) and \( P(H = h) \) iteratively, and further to refine the predictive model \( M \) in Eq. (10). By the iteration of effort prediction for projects with unknown effort and refinement for predictive model, EM is integrated with Bayesian regression for effort prediction. Following this line of thought, we propose BREM (Bayesian regression with EM) to predict software effort. The procedures of BREM are very similar to those employed in Bayesian regression except the adoption of iterative update of \( P^{(\theta + 1)}(X_j = x, H = h) \), \( P^{(\theta + 1)}(H = h|x_0) \) and \( P^{(\theta + 1)}(H = h) \). We describe its procedures in details as follows.

First, for a numeric attribute, its likelihood \( P^{(\theta + 1)}(X_j = x | H = h) \) at the \( \theta + 1 \) iteration is kept as the same as Eq. (5). However, the joint probability \( P^{(\theta + 1)}(X_j = x, H = h) \) at the \( \theta + 1 \) iteration should be adapted to Eq. (12).

\[
P^{(\theta + 1)}(X_j = x, H = h) = \frac{1}{(m + m') b_{X_j}} \sum_{i=1}^{m+m'} K \left( \frac{x - x_i}{b_{X_j}} \right) K \left( \frac{h - h_i}{b_{h}} \right)
\]  

(12)

At the first iteration, \( b_{X_j} \) is estimated with projects merely from the training dataset. For the remaining iteration \( (\theta \geq 1) \), \( b_{X_j} \) is estimated with projects from both training set with real effort and test set with predicted effort in the \( \theta - 1 \) iteration. That is, \( b_{X_j} = 1.06 \sigma_0 (m + m')^{-1/2} \) where \( \sigma_0 \) is estimated from the effort of both training and test projects and \( m' \) is the number of test projects.

For a categorical attribute, its likelihood \( P^{(\theta + 1)}(X_j = x | H = h) \) at the \( \theta + 1 \) iteration is kept as the same as Eq. (8). However, the likelihood of \( P^{(\theta + 1)}(H = h | x_0) \) should be adapted to Eq. (13) where \( b_{X_j|x_0} \) is the new bandwidth for effort of both training and test projects whose values under the attribute \( X_j \) are equal to \( x_0 \).

\[
P^{(\theta + 1)}(H = h | x_0) = \frac{1}{P(x_0)} \sum_{q=1}^{m|x_0|} K \left( \frac{h - h_q}{b_{h|x_0|}} \right)
\]  

(13)

Second, the prior probability \( P^{(\theta + 1)}(H = h) \) at the \( \theta + 1 \) iteration is estimated using Eq. (14).

\[
P^{(\theta + 1)}(H = h) = \frac{1}{(m + m') b_{h}} \sum_{i=1}^{m+m'} K \left( \frac{h - h_i}{b_{h}} \right)
\]  

(14)

Third, the predictive model at the \( \theta + 1 \) iteration should be adapted accordingly as shown in Eq. (15) where \( G^0 \) is the set of renewed grid points using both training and test projects with the same definition in Eq. (10).

\[
\sum_{h \in G^0} P^{(\theta + 1)}(H = h | D_{test}) > 0.5
\]  

(15)

Here, the joint probability \( P^{(\theta)}(H = h | D_{test}) \) is estimated using Eq. (16). Thus, for a test project \( D_i \), its effort is predicted by BREM as \( h_{D_i} \).

\[
P^{(\theta)}(H = h, D_{test}) = P^{(\theta)}(H = h) \prod_{j=1}^{n} P(X_j = x_j | H = h)
\]  

(16)

BREM attempts to maximize a posteriori estimate of the software projects with their effort, i.e. \( P(D) = [P(D_{test}|D_{train}) \) (\( D_{test} \) is the set of test projects and \( D_{train} \) is the set of training projects). The iteration of EM algorithm will terminate at the \( \theta^* \) iteration when \( P^{(\theta')}(D) \) represented by Eq. (17) is maximized.

\[
P^{(\theta)}(D) = \prod_{D_{test} \not\in \{ D_{train} \}} \sum_{h \in G^0} P^{(\theta)}(h | D_{test}) P^{(\theta)}(D_{train} | h) \prod_{D_{test} \in \{ D_{train} \}} P(h | D_{test})
\]  

(17)

We will work with log \( P^{(\theta)}(D) \), instead of trying to maximize \( P^{(\theta)}(D) \) directly because there are too much small decimals in \( P^{(\theta)}(D) \) resulting in unattainable computation. In the E-step, we update the probabilities of \( P^{(\theta)}(X_j = x_j | H = h) \) (Eqs. (12) and (13)) and \( P^{(\theta)}(H = h | D_{test}) \) (Eq. (14)) iteratively. In the M-step, we calculate \( P^{(\theta)}(D) \) (Eq. (17)). For space limitation, we will not prove the correctness of EM algorithm with Bayesian regression mathematically. One of interest can refer to Minka [21] and Nigam et al. [25] for more details.

4. Embedded missing handling strategies in BREM

When Eqs. (5) and (8) are used to estimate the likelihood of \( X_0 = x \) with respect to effort \( h_i \), i.e. \( P(X_0 = x | h) \), we do not consider missing values involved in \( x_0 (1 \leq i \leq m) \). This section will propose two strategies as MDT and MDI to handle missing values when using BREM for effort prediction.

For each attribute \( X_j \), we can divide the whole historical dataset \( D \) into two subsets, i.e. \( D = [D_{obs} | D_{mis}] \) where \( D_{obs} \) denotes the set of projects whose values on attribute \( X_j \) were observed and \( D_{mis} \) denotes the set of projects whose values on attribute \( X_j \) were unobserved. Note that both \( D_{obs} \) and \( D_{mis} \) contain both training and test projects.

We may also divide the attributes of a project \( D \) into two subsets, i.e. \( X = [X_{obs} | X_{mis}] \) where \( X_{obs} \) denotes the set of attributes whose values are observed in project \( D \) and \( X_{mis} \) denotes the set of attributes whose values are missing for project \( D \).

With the partition of \( D \) and \( X \), we propose two missing data handling strategies as the Missing Data Toleration (MDT) strategy and the Missing Data Imputation (MDI) strategy to handle missing data in historical dataset when BREM is used for software effort prediction.

4.1. Missing data toleration strategy

This strategy is very similar with the method adopted by C4.5 to handle missing data. Under this strategy, we ignore the missing values in training the effort prediction model. Note that this strategy is different with LD (List Deletion) technique that deletes all the projects containing missing values. To estimate \( P^{(\theta + 1)}(X_j = x | h) \),
for a numeric attribute $X$, we rewrite Eq. (12) into Eq. (18) as follows and keep Eq. (5) as the same.

$$p^{(i+1)}(X = x, H = h) = \frac{1}{|D_{obs}|} \sum_{k=1}^{|D_{obs}|} \sum_{i=1}^{|D_{obs}|} K \left( \frac{x - x_k}{b_{x_k, h_k} h_k} \right) \left( \frac{h - h_k}{b_{h_k, h_k}} \right)$$

(18)

Here, $b_{x_k, h_k}$ is the bandwidth of values on the attribute $X$ of projects in $D_{obs}$ and $b_{h_k, h_k}$ is the bandwidth of effort of projects in $D_{obs}$. For consistency, the prior probability $p^{(i+1)}(H = h)$ is estimated using Eq. (19).

$$p^{(i+1)}(H = h) = \frac{1}{|D_{obs}|} \sum_{k=1}^{|D_{obs}|} K \left( \frac{h - h_k}{b_{h_k, h_k}} \right)$$

(19)

For a categorical attribute $X$, to estimate the likelihood $p^{(i+1)}(H = h | X = x_k)$, we rewrite Eqs. (13)-(20) merely considering the projects in $D_{obs}$.

$$p^{(i+1)}(H = h | X = x_k) = \frac{1}{P_{obs}} \sum_{j=1}^{P_{obs}} K \left( \frac{x - x_{sj}}{b_{x_{sj}, h_{sj}} h_{sj}} \right)$$

(20)

Here, $i(x_k, x_{sj}, D_i \in D_{obs})$, returns the number of projects in $D_{obs}$ whose values on $X$ is equal to $x_k$.

$$p^{(i+1)}(X = x_k, H = h) = \frac{p^{(i)}(X = x_k, H = h | X = x_k, D_i \in D_{obs})}{|D_{obs}|}$$

(21)

$$p^{(i+1)}(X = x_k | H = h) = \frac{p^{(i)}(X = x_k) p^{(i)}(H = h | X = x_k)}{\sum_{q=1}^{P_{obs}} p^{(i)}(X = x_q) p^{(i)}(H = h | X = x_q)}$$

(22)

Under this strategy, the predictive model $M$ for test projects $D_t$ ($1 \leq i \leq m'$) is kept as the same as Eq. (15).

4.2. Missing data imputation strategy

This strategy attempts to use both observed values and missing values in historical dataset to train the predictive model $M$. The basic idea of the MDI strategy is that unobserved values can be imputed using observed values. Then both observed values and imputed values are used to construct the predictive model. The MDI strategy is an embedded processing in BREM.

To estimate $p^{(i+1)}(X = x | h)$, for a numeric attribute $X$, the Eq. (12) should be adapted to Eq. (23) to estimate the joint probability $p^{(i+1)}(X = x, H = h)$ and Eq. (5) is kept as the same to estimate $p^{(i+1)}(X = x | H = h)$, $b_{x_k}$ is the bandwidth of both observed values and imputed values of the attribute $X$ at the $\theta$ iteration. $b_{h_k}$ is the bandwidth of effort of both training and test projects at the $\theta$ iteration.

$$p^{(i+1)}(X = x, H = h) = \frac{1}{(m + m') b_{x_k} b_{h_k}} \sum_{i=1}^{m + m'} K \left( \frac{x - x_i}{b_{x_i}} \right) \left( \frac{h - h_i}{b_{h_i}} \right)$$

(23)

The missing value $x_m$, which is the value of project $D_i \in D_{mis}$ under the attribute $X$, is imputed using $x_m$ with Eq. (24).

It can be seen from Eq. (24) that $x_m$ is a constant independent of $D_i$, given its effort $h_i$. If $D_i$ comes from the training set, then its effort $h_i$ is known. Otherwise, if $D_i$ comes from the test set, then its effort $h_i$ is predicted using Eq. (15). For the likelihood of $h_i$ with respect to the project $D_i$, i.e. $p^{(i)}(D_i | h_i)$, it can be estimated using Eq. (25).

$$p^{(i)}(D_i | h_i) = \prod_{j=1}^{n} p^{(i)}(X = x_j | h_i)$$

(25)

For a categorical attribute $X$, we impute its missing value $x_m$ in project $D_i$ straightly using $x_{m}^{(i+1)}$ shown in Eq. (26). That is, to impute the missing value using the value from the project that the effort $h_i$ is mostly likely with respect to. Then Eqs. (13), (14) and (8) can be used to estimate the likelihood $p^{(i+1)}(X = x_m | h_i)$. In the first iteration, all the likelihood and priors are estimated using merely projects in the training set.

$$x_{m}^{(i+1)} = \arg_{x_m} \max_{x_m} \prod_{j=1}^{n} p^{(i)}(X = x_j | h_i)$$

(26)

4.3. Discussion

It can be seen from Eq. (18) that the MDT strategy makes use of more information of observed values in historical dataset than traditional LD method. Those observed values of come from both training and test projects. This strategy is inspired by the full information maximum likelihood (FIML) in classic statistics in handling incomplete data. However, it is different from FIML in that the latter does not take the model structure of Bayesian regression into account. The basic assumption underlying the MDT strategy is that, if the missing data is MAR, information contained in observed values is enough to construct the predictive model. The bias in the model would not be enlarged because it kept the difference of values under same attributes and, computation complexity is also reduced under the MDT strategy because, it does not need additional computation.

The basic idea of the MDI strategy is to use $D_{obs}$ to impute the missing value of $D_{mis}$. Simply, we can regard Eq. (24) as expectation substitution. That is, to use the expectation of $x_m$ to impute the missing value. It is different with mean substitution because $x_m$ is dependent on $p^{(i)}(D_i | h_i)$. In this sense, it is much more like Class Mean Imputation (CMI), especially for training projects. The concept of observed $x_m$ to $x_m$ depends on the probability $p^{(i)}(D_i | h_i)$, which gauges the likelihood of a project with effort $h$, being $D_i$. This likelihood is completely decided by specific $h_i, D_i$ and the Bayesian regression model in the $\theta$ iteration. This mechanism can be explained as multiple imputation based on project weighting, that has not ever mentioned in existing missing handling techniques.

In an extreme case where there is no missing value of the attribute $X$, i.e. $D_{mis} = \phi$, then Eq. (23) degenerates to Eq. (18). The difference between the MDT and MDI strategies is in that the MDI use $x_m$ to smooth the distribution of $p^{(i+1)}(X = x, H = h)$ as can be seen from Eqs. (18) and (23). The motivation of introducing test projects in constructing predictive model $M$ is to reduce the variances of parameters in Bayesian regression. In theory, the variances of those parameters are further reduced if missing values are imputed and considered in the MDI strategy.

The values of $x_m$ is updated at each iteration of EM when $p^{(i)}(D_i | h_i)$ is updated by test projects. The MDI strategy would not
hurt the distribution of values of attributes, which is the advantage inherited from mean imputation. Meanwhile, the variance of values of attributes would be augmented by imputing a missing value with different values at different iterations. This point is very similar with Bayesian multiple imputation (Schafer and Graham, 2002), which imputed a missing value with multiple values in order to augment the variance. In machine learning, the expected error is equal to variance plus bias [5]. The goal to augment variance is an indirect method to reduce bias among values of attributes of projects with different effort.

Friedman [10] propose Bayesian structural EM algorithm (Bayesian-SEM) to cope with the incomplete data in training Bayesian belief network. The basic idea there is to approximate the probability $P(H|M, D) \approx P(H|M', \Theta)$ for missing data in $D$ where $\Theta$ is a set of parameters for the Bayesian model. That is, Bayesian-SEM approximates the likelihood of missing values $P^{\theta}(x_{ij} | D_i, \Theta)$ under Bayesian model and observed data using $P^{\theta}(x_{ij} | D_i, \Theta) = (D_i \in D_{obs})$. In this sense, we admit that both MDT and MDI strategies are actually specific implementations of Bayesian-SEM to consider the model and data simultaneously in posterior probability approximation, and to elaborate parameters $\Theta$ iteratively. However, the MDI strategy goes further than Bayesian-SEM in that it imputes a real value for the missing data while Bayesian-SEM merely approximates its posterior probability. That is, all the estimates in Bayesian-SEM is purely based on observed data (very similar to the MDT strategy) while we also consider the imputed values of missing values in the MDI strategy as shown in Eqs. (20), (21) and (23).

5. Experiments

5.1. Research questions

To examine performances of BREM for software effort estimation and the two proposed strategies for handling missing data, we conduct a series experiments to answer the following two research questions.

RQ1. Is the performance of BREM better than existing learners in software effort estimation?

RQ2. How are the performances of the MDT and MDI strategies in handling missing data when BREM is used for software effort estimation?

To answer RQ1, we introduce LR (Linear Regression), BR (Bayesian Regression), SVR (Support Vector Regression), M5’ regression tree [27] and CBR (Case-Based Reasoning) for performance comparison. The kernel method used in SVR is linear kernel because it produced better performance than other kernels in applications such text classification and image segmentation [43]. LR is widely used in software engineering for predicting effort and defects [6]. We introduce M5’ regression tree for comparison because C4.5 is capable of tolerating missing values and M5’ is one of its variants. We introduce CBR because it is widely used in software effort estimation [34]. We use the existing packages from Weka\(^1\) platform to implement all the methods mentioned in the experiments.

To answer RQ2, we introduce the independent missing imputation techniques including MI, BMI, CMI, and MINI for comparison. On the one hand, we use BREM embedded with the MDT and MDI strategies for effort estimation using the training and test data directly. On the other hand, we employ the independent imputation techniques to impute the missing values and then BREM is used for effort estimation using the imputed values. We also observe the difference of the imputed values and real values measured by relative error (RE) described in Eq. (27).

We use the mode to impute a missing value under categorical attributes when MI is adopted. The values of numeric attributes are characterized using multivariate normal distribution when BMI is adopted. We also conduct a goodness of fit test of multivariate normal distribution for values of the numeric attributes. For each categorical attribute, we simply use its probability mass function derived from observed values to simulate its value distribution. We initialize BMI using the FIML method and execute BMI 15 times in parallel with the observed data and each execution has 15 iterations. Then, each missing value is imputed by average of the 15 samples from the 15 parallel chains at the final iteration.

RE = \[
\begin{cases} 
\frac{\sum x_i - \bar{x}}{\sigma_x} , & \text{if } X_i \text{ is a numeric attribute;} \\
\frac{\sum_{x_i \in \text{nominal values}} x_i - x_{\text{most common}}}{\text{number of nominal values}}, & \text{if } X_i \text{ is a nominal attribute.}
\end{cases}
\] (27)

For a project with missing values, when CMI and MINI are used, we regard those projects whose effort is within 10% deviation of the project effort as having the same class. We admit that despite this disposal is a little bit arbitrary, but there is no well recognized method to classify project effort into different categories as they are real numeric values.

5.2. The datasets

The datasets we used are from the ISBSG and CSBSG databases. The ISBSG database\(^2\) contains 1238 projects from insurance, government, etc., of 20 different countries and, each project is described with 70 attributes. We only use the projects whose “Data Quality Rating” is equal or above “B” in our experiments. We prune the ISBSG dataset into 136 projects with 28 attributes by the criterion that each project and attribute should include no missing values. Among the 28 attributes, 22 of them are categorical attributes and 6 of them are continuous attributes.

The CSBSG database contains 1103 projects from Chinese software industry. It is created in 2006 with its mission to promote Chinese benchmarking standards of software productivity. Those projects are collected from 140 organizations and 15 regions across China by Chinese association of software industry [12]. Each project is described by 179 attributes. We adopt the same criterion as used in ISBSG data set to prune the CSBSG data set into 117 projects and 34 attributes, which includes 14 categorical attributes and 20 continuous attributes. One of interest in these two datasets can see Zhang et al. [41] for more details of the attributes.

5.3. Study of RQ1

We set a parameter as percentage of test projects range from 0.05 to 0.5 with intervals as 0.05 to investigate different methods on effort prediction. For instance, if the percentage of test projects is set as 0.1, then 13 projects will be randomly selected from ISBSG dataset to examine the performances of different methods mentioned above. The upper bound of the percentage of test projects is set as 0.5 because we hold that too many test projects are meaningless for software effort prediction in real practice. Leave-one-out cross-validation [44,17] is used to simulate real practice of effort estimation. We repeat each experiment 10 times and the performance is averaged on the 10 repetitions. For instance, when 10% of projects are used in training, we only use one project randomly sampled from the remaining 90% of projects for test at one repetition.

\(^1\) Online: http://www.cs.waikato.ac.nz/ml/weka/.

\(^2\) Online: http://www.isbsg.org.
Fig. 1 shows the performance of BREM compared with LR, BR, SVR and M5' in predicting effort of the ISBSG and CSBSG projects. We can see that BR and BREM produce better performances than other existing learners when the percentage of test projects is small (not larger than 0.3 for the ISBSG dataset and not larger than 0.25 for the CSBSG dataset). We explain this outcome as that BR and BREM have superiority over other methods if the number of training projects is larger than the number of test projects. When more and more test projects are appended into the task, all the examined methods are unable to characterize the relationship between attributes and software effort. When a small number of test projects are appended into BREM, these projects can actually reduce variances of the parameters of BR. However, when then number of test projects is large, the increasing uncertainness induced by test projects, which also accounts for the low performances of other methods, deteriorates BREM and results in its low performances.

To better illustrate the effectiveness of each effort prediction method, the non-parametric Wilcoxon signed-rank test is employed to analyze the differences of MREs (see Eq. (2)) produced by each method. Tables 2–5 show the results of Wilcoxon signed-rank test of the MREs of the six methods in predicting effort of ISBSG and CSBSG projects. For each dataset, we investigate two scenarios: one is when the percentage of test projects is not larger than 0.30 and another is when the percentage of test projects is larger than 0.30. The reason here is that we see from Fig. 1 that the results of performance comparison of the six methods are different between before and after 30% of test projects. In the first scenario, we examine the differences of 60(6 × 10) pairs of MREs from each pair of the six methods and in the second scenario, we compare 40(4 × 10) pairs of MREs from each pair of the six methods.

The following codification of the $P$-value in ranges was used: “$\gg$” and “$\ll$” mean that the $P$-values is lesser than or equal to 0.01, indicating a strong evidence of that a method generates a greater or smaller MRE than another one, respectively; “$\ll$” and “$\gg$” mean that $P$-value is bigger than 0.01 and minor or equal to 0.05, indicating a weak evidence that a method generates a greater or smaller MRE than another one, respectively; “$\sim$” means that the $P$-value is greater than 0.05 indicating that it does not have significant differences in the MREs of the methods.

From Tables 2–5, we can also see that when the percentage of test projects is not larger than 0.30, BREM and BR have produced significantly better performances than other methods in both ISBSG and CSBSG datasets. However, it is not the case when the percentage of test projects is larger than 0.30. Thus, our answer for RQ1 is quite straightforward. BREM can produce better performances than other methods investigated in the experiments under the condition that the number of test projects is relatively smaller.
than that of training projects. In the best case, BREM can improve \( \text{PRED}(25) \) at ratio as 19.82% compared with LR using 25% of projects for test in the ISBSG dataset. In the worst case, BREM has similar performance to other effort prediction methods. Considering that in real practice, the number of projects needed to predict its effort is much smaller than the number of existing projects with known effort, we argue that the BREM is of practical use for software organization in effort prediction.

### 5.4. Study of RQ2

We use 2% of test projects to study RQ2. The first percentage is 15% because, we see from Fig. 1 that BR and BREM have produced stable performances around this percentage in both ISBSG and CSBSG datasets. The second percentage is 35% because we see from Fig. 1 that the performances of BR and BREM are not better than those of other methods at this percentage. Another parameter we set for the experiments is the level of missing data that ranges from 0.025 to 0.25 with intervals as 0.025. To make the experimental results comparable, we only add missing values into the training dataset. For instance, if we set the level as 0.025, then it means that for each attribute, 2.5% of its values in training set will be replaced with missing values randomly. We argue that too much missing data in the experiments will cause all the methods incapable of missing handling and we also validate this point in our previous work [41]. We do not replace any values in test set with missing values for fair comparison. We also repeat each experiment 10 times with leave-one-out cross-validation [44,17] and the performance is averaged on the 10 repetitions.

Fig. 2 shows the performances of BREM with the MDT and MDI strategies in software effort prediction with missing data using 15% of test projects. We can see that the MDT and MDI strategies produce significantly higher \( \text{PRED}(25) \) than other methods. When the level of missing data is becoming larger, which means more missing values are appended to the training set, the performances of all the methods are deteriorating. However, we can see that BREM with the MDT and MDI strategies outperform other methods in both ISBSG and CSBSG datasets. The independent techniques, though can be combined with arbitrary predictive methods, are not comparable to the MDT and MDI strategies embedded with BREM in handling missing data in effort prediction.

We also can see that when the level of missing data is no larger than 0.05, the MDT strategy outperforms the MDI strategy in both ISBSG and CSBSG datasets. We explain that when there is small proportion of missing values, the observed data is enough to construct a favorable predictive model for effort prediction. However, when the proportion of missing values increases, the MDI strategy, that utilizes imputed values for missing data, outperforms the MDT strategy.

To better illustrate the effectiveness of each method for effort prediction with missing data, we also employ the non-parametric Wilcoxon signed-rank test to directly analyze the MREs produced by each pair of methods. Tables 6 and 7 show the results of Wilcoxon signed-rank test of the MREs of the seven methods in predicting effort of ISBSG and CSBSG projects, respectively, when the percentage of test projects is set as 0.15. The following codification of the \( P \)-value in ranges was used: ‘\( > \)' and ‘\( < \)' mean that the \( P \)-values is lesser than or equal to 0.01, indicating a strong evidence of that a method generates a greater or smaller MRE than another one, respectively; ‘\( \leq \)' and ‘\( > \)' mean that \( P \)-value is bigger than 0.01 and minor or equal to 0.05, indicating a weak evidence that a method generates a greater or smaller MRE than another one, respectively; ‘\( = \)' means that the \( P \)-value is greater than 0.05 indicating that it does not have significant differences in the MREs of the methods.

![Fig. 2. The performances of BREM compared with M5', and BR combined with independent missing imputation techniques using 15% of test projects.](image-url)
is set as 0.35. The following codification of the $P$-value in ranges was used: "\textless" and "\geq" mean that the $P$-values is lesser than or equal to 0.01, indicating a strong evidence of that a method generates a greater or smaller MRE than another one, respectively; "<" and "\geq" mean that $P$-value is bigger than 0.01 and minor or equal to 0.05, indicating a weak evidence that a method generates a greater or smaller MRE than another one, respectively; "\geq" means that the $P$-value is greater than 0.05 indicating that it does not have significant differences in the MREs of the methods.

We can see from Table 8 that in ISBSG dataset, BREM with MDT and MDI strategies outperform other methods. However, this is not the case when we see Table 8 that BREM with MDT only outperforms BR with MI with strong evidence. We explain that when we set the percentage of test projects as 0.35, BREM has similar performance with BR and M5\textsuperscript{0} in ISBSG dataset but there are large performance differences compared with other methods in CSBSG dataset (see Fig. 1). However, BREM with the MDI strategy always outperforms other methods in both ISBSG and CSBSG datasets.

We explain the outcome that expectation imputation used in the MDI strategy takes effects in most cases when there is a large proportion of missing values in constructing the predictive model. The MDI strategy informs the predictive model that there is a missing value in the training process and attempts to compensate that information loss to the maximum length by making use of available information. For the MDT strategy, it is a lazy strategy and ignores the missing values and produces a predictive model by using fully observed information. The model “knowing” that there are missing values comes up with superiority than just “ignoring” the missing values.

Fig. 4 shows the performances, measured by RE as defined in Eq. (27), of the MDI strategy and other independent missing imputation techniques in imputing missing values, when BREM and other methods are employed to predict software effort. We can see that when the level of missing data is not larger than 0.15, the MDI strategy produces significantly better performances than other independent missing imputation techniques (Wilcoxon signed-rank test compared with BMI, in ISBSG dataset, $P = 0.0012$; in CSBSG datasets, $P = 0.0023$). We explain the outcome that when there is small missing data in training set, the better performance of BREM (see Fig. 2) makes the likelihood $P(h|D,h_s)$ more precise estimates than that with large missing data. As a result, for each missing value, the expectation used for imputation in the MDI strategy is more approximately approaching to its real value.

From the above analysis, our answer for RQ2 can be stated as follows. When BREM is employed for software effort prediction
with missing data, with the MDT strategy to handle missing data, on the condition that BREM has better or similar performance with other methods without missing data, it produces better performances than Bayesian regression with independent missing imputation techniques and M5. However, with the MDI strategy, BREM can always outperform other methods in handling missing data in software effort prediction. Nevertheless, when the missing data in training dataset is small, the imputations provided by the MDI strategy are significantly accurate than those provided by independent missing imputation techniques.

5.5. Threats to validity

The threats to external validity primarily include the extent to which the attributes we use to describe the projects and its representative capacity of the ISBSG and CSBSG datasets. In Section 2, we assume that missing mechanism for software effort datasets is under MAR and we can transfer MNAR into MAR by some means. Moreover, as mentioned in Section 3.1, Bayesian regression assumes attribute independent of each other that are mostly in conflict with real practice. These threats could be reduced by statistical test and feature selection in future work. In Section 5.3, we see from Fig. 1 that if the percentage of test projects becomes larger than a critical percentage (0.30 for ISBSG dataset and 0.25 for CSBSG dataset), then BREM cannot outperform other methods. This is because the prediction model learned from the historical data (i.e. the training data) cannot be applied to test projects as the lack of enough historical data. We admit that the critical percentage should be varied for different dataset. To generalize BREM to other datasets, we need to ensure that there is enough data for training. On the one hand, this is not a problem if an organization collects their projects’ effort intentionally and adds those collected projects into historical dataset continuously, there should be enough historical dataset for training. On the other hand, we see from Fig. 1 that the better is the performance of BREM with smaller percentage of test projects. However, if a large number of projects need to predict their effort, then a conservative way to use BREM is to predict the incoming projects one by one chronologically. After each prediction, we add the project with predicted effort into training dataset and use the project and other existing projects to train a new predictive model to predict effort of next project. By this way, we can ensure the proportion of the size of training data to the size of test data is as small as possible. In Section 5.4, we see from Fig. 4 that if the level of missing data is not larger than

<table>
<thead>
<tr>
<th>Methods</th>
<th>BR + CMI</th>
<th>M5</th>
<th>BR + MINI</th>
<th>BR + BMI</th>
<th>BREM + MDT</th>
<th>BREM + MDI</th>
</tr>
</thead>
<tbody>
<tr>
<td>BR + MI</td>
<td>~</td>
<td>&gt;</td>
<td>&gt;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BR + CMI</td>
<td></td>
<td>&gt;</td>
<td>&gt;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M5</td>
<td></td>
<td>~</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BR + MINI</td>
<td></td>
<td>&gt;</td>
<td>~</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BR + BMI</td>
<td></td>
<td>&gt;</td>
<td>~</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BREM + MDT</td>
<td></td>
<td>&gt;</td>
<td>&gt;</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 8
Results of Wilcoxon signed-rank test of the seven methods in predicting effort of ISBSG projects when the percentage of test projects is set as 0.15.

<table>
<thead>
<tr>
<th>Methods</th>
<th>BR + CMI</th>
<th>M5</th>
<th>BR + MINI</th>
<th>BR + BMI</th>
<th>BREM + MDT</th>
<th>BREM + MDI</th>
</tr>
</thead>
<tbody>
<tr>
<td>BR + MI</td>
<td>~</td>
<td>&gt;</td>
<td>&gt;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BR + CMI</td>
<td></td>
<td>&gt;</td>
<td>&gt;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M5</td>
<td></td>
<td>~</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BR + MINI</td>
<td></td>
<td>&gt;</td>
<td>~</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BR + BMI</td>
<td></td>
<td>&gt;</td>
<td>~</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BREM + MDT</td>
<td></td>
<td>&gt;</td>
<td>&gt;</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 9
Results of Wilcoxon signed-rank test of the seven methods in predicting effort of CSBSG projects when the percentage of test projects is set as 0.15.

<table>
<thead>
<tr>
<th>Methods</th>
<th>BR + CMI</th>
<th>M5</th>
<th>BR + MINI</th>
<th>BR + BMI</th>
<th>BREM + MDT</th>
<th>BREM + MDI</th>
</tr>
</thead>
<tbody>
<tr>
<td>BR + MI</td>
<td>~</td>
<td>&gt;</td>
<td>&gt;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BR + CMI</td>
<td></td>
<td>&gt;</td>
<td>&gt;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M5</td>
<td></td>
<td>~</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BR + MINI</td>
<td></td>
<td>&gt;</td>
<td>~</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BR + BMI</td>
<td></td>
<td>&gt;</td>
<td>~</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BREM + MDT</td>
<td></td>
<td>&gt;</td>
<td>&gt;</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig. 4. The performances of the MDI strategy on imputing missing values compared with independent missing imputation techniques.
10%. MDI can produce more accurate imputation than other techniques. We admit that this result is observed from only ISBSG and CSBSG datasets at this time. The accuracy of imputed values of MDI is decided by Eq. (24). If too much missingness is involved in the training dataset, then the distribution of \( h_i \) will be hurt seriously leading to inaccurate imputation for \( x_{ni} \). Moreover, when using BEM for dataset with missing level larger than 10%, we can use an incremental way to impute the values under the attribute with smallest missingness firstly and then use the imputed values to impute next attribute with second smallest missingness. This procedure can be used iteratively by analogy until all the missing values are imputed.

The threats to internal validity are measurement and data effects that can bias our results.Faults in cross-validation technique, the disadvantage of PRED(25) for measuring performance of effort prediction, and the moderate size of the datasets used in the experiments might cause such effects. To reduce these threats, we will attempt more datasets of software effort and introduce various kinds of performance measures. One threat to construct validity is that our experiment makes use of clipping attributes and clipping project data from both ISBSG and CSBSG datasets, we hope that these manipulations will still precisely capture the internal characteristics of software attributes and its effort.

6. Concluding remarks and future work

This paper proposes BREM for software effort prediction and two embedded strategies for handling missing data. Inspired by the formulation of Bayesian regression and its superiority over other predictive models in machine learning, we combine Bayesian regression and EM algorithm to make full use of training and test data in constructing the predictive model. Further, to make use of incomplete data, we propose the MDT and MDI strategies embedded with BREM to handle missing data in software effort prediction.

The contribution of this paper can be summarized in two manifolds. First, we customize Bayesian regression and EM algorithm for software effort prediction. Naïve Bayes is usually adopted in classification and seldom used in it regression. With the experiments on ISBSG and CSBSG datasets, we validate the effectiveness of BREM for software effort prediction in comparison with LR, BR, SVR and M5’. Naïve Bayes is behind by posterior probability theory and EM algorithm is supported by maximum likelihood estimates of parameters in probabilistic models using incomplete data.

Although recently naïve Bayes and EM algorithm has already been used in machine learning tasks such as text classification [25], we believe that this paper is the first one to combine Bayesian regression and EM algorithm in software effort prediction.

Second, we propose two strategies to handle missing data when using BREM for software effort prediction. The MDT strategy simply ignores missing values and makes full use of observed values of software projects. It has the advantage of small computation complexity. The MDI strategy uses observed values of attributes to impute missing values. Our imputation strategy is different from other imputation techniques in that the imputed values are effort dependent and iteratively updated. That is, for different effort of projects, its missing values should be imputed using different values. We provide a theoretical analysis for these two strategies and conduct extensive experiments to examine the performances of the proposed strategies in comparison with BR and other independent imputation techniques for software effort prediction. In particular, we compare the relative error of the imputed values given by the MDI strategy and other independent strategy. The experimental results validate the superiority of the MDI strategy.

In the future, we will conduct more experiments to investigate the performances of the MDT and MDI strategies in handling missing data when BREM is used for predictive tasks. Also, we will attempt to develop more techniques dealing with missing data under the framework of BREM with the goal of improving accuracy of software effort prediction.

Acknowledgement

This research was supported in part by National Natural Science Foundation of China under Grant Nos. 71121007, 61173041, 61273041 and 61432001; Beijing Natural Science Fund under Grant No. 41120087. The authors of the paper would like to appreciate the anonymous reviewers for their valuable comments.

References