Neurocomputing 73 (2010) 3039–3065

Contents lists available at ScienceDirect

Neurocomputing

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

A neural network-based framework for the reconstruction of incomplete data sets

Iffat A. Gheyas *, Leslie S. Smith

University of Stirling, Department of Computing Science and Mathematics, Stirling, Scotland FK9 4LA, UK

ARTICLE INFO

Article history:
Received 20 May 2009
Received in revised form
18 March 2010
Accepted 28 June 2010
Communicated by B. Apolloni
Available online 10 August 2010

Keywords:
Missing values
Imputation
Single imputation
Multiple imputation
Generalized regression neural networks

ABSTRACT

The treatment of incomplete data is an important step in the pre-processing of data. We propose a novel nonparametric algorithm Generalized regression neural network Ensemble for Multiple Imputation (GEMI). We also developed a single imputation (SI) version of this approach—GESI. We compare our algorithms with 25 popular missing data imputation algorithms on 98 real-world and synthetic datasets for various percentage of missing values. The effectiveness of the algorithms is evaluated in terms of (i) the accuracy of output classification: three classifiers (a generalized regression neural network, a multilayer perceptron and a logistic regression technique) are separately trained and tested on the dataset imputed with each imputation algorithm, (ii) interval analysis with missing observations and (iii) point estimation accuracy of the missing value imputation. GEMI outperformed GESI and all the conventional imputation algorithms in terms of all three criteria considered.

1. Introduction

Incomplete data is an unavoidable problem when dealing with real world datasets. Missing values result in less efficient estimates because of the sample bias, and the reduced sample size. Further, most data mining algorithms cannot work directly with incomplete datasets. To make the matter worse, many real world problems are high dimensional. If missing data are randomly distributed across cases, we could even end up with no valid cases in the dataset, because each of them will have at least one missing data element. Hence, missing value imputation is widely used, by necessity. Imputation refers to the replacement of missing data with statistically plausible values. However, a naïve or unprincipled imputation may create more problems than it solves. A poor imputation strategy may result in distorted created samples that can mislead classification, prediction and clustering techniques. The algorithm used to generate imputed values must be "correct", that is, it must accommodate the necessary predictor variables and their associations. Rubin contends that good imputation methods use all information related to missing cases [1].

Missing values occur when values are not recorded for all attributes. Little and Rubin [2] and Schafer [3] classify non-random missing data into three categories: (i) missing completely at random (MCAR), (ii) missing at random (MAR), and (iii) missing not at random (MNAR). MCAR occurs when the data which is missing does not depend on the values of any variable in the dataset. Possible reasons for MCAR include manual data entry procedure, incorrect measurements, equipment error, changes in experimental design, accidental skipping of a question or questions, etc. MAR occurs when the probability of missing data on a particular variable x_1 (say, smoking status) depends on other observed variables (x_2,x_3,...,x_n) (say, age), but not on x_1 itself. For example, teenagers are less likely to answer a question on smoking habits. MNAR occurs when the probability of missing data on particular variable x_1 depends on the variable x_1 itself. For example, a question regarding smoking habits is less likely to be answered when the respondent is a smoker. MCAR and MAR data are recoverable, whereas MNAR is not.

Missing data imputation is challenging because possible biases exist since the subjects with missing values are often systematically different from the subjects without missing values [1]. These biases are difficult to eliminate since the precise reasons for missing data are usually not known. Hence, imputations should reflect the full uncertainty about missing values. However, the determination of uncertainty is not straightforward. If the uncertainty is underestimated, the classifier trained with the

Abbreviations: EM, expectation maximization; GA, genetic algorithm; GRNN, generalized regression neural networks; HD, hot-deck imputation; HUX, half uniform crossover; KNN, K-nearest neighbours; MAR, missing at random; MCAR, missing complete at random; MCMC, Markov chain Monte Carlo; MI, multiple imputation; MLP, multilayer perceptrons; MNAR, missing not at random; MS, mean substitution; PCA, principal component analysis; PSO, particle swarm optimization; RBFN, radial basis function networks; SA, simulated annealing algorithm; SI, single Imputation; WKNN, weighted K-nearest neighbours; ZI, zero imputation

* Corresponding author.
E-mail address: iag@cs.stir.ac.uk (I.A. Gheyas).

0925-2312/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2010.06.021

imputed dataset will overfit the training data and produce erroneous outputs. To fully account for all sources of variation, it is essential to allow for sampling variation and imputation variation in the imputation. Sampling variation occurs when we sample a population and estimate a parameter from the sample rather than from the population of interest as a whole. If we would have taken a different sample from the population, we would have obtained different parameter estimates. Imputation variation is similar. Missing values are replaced by the best surrogate values. However, an imputed value is just a guess at the actual value; it is not an actual (observed) value. Imputation variation arises from the fact that there is uncertainty regarding the actual value of the missing data. Imputation can also lead to an overestimation of the uncertainty of the estimate. If the uncertainty is overestimated, the classifier trained with the imputed dataset will underfit the training data and exhibit poor prediction capability. Procedures for imputation that incorporate appropriate variability among imputations within a model are called "proper" [1].

A major focus of research is to develop an imputation algorithm that preserves the multivariate joint distribution of input and output variables. Much of the information in these joint distributions can be described in terms of means, variances and covariances. If the joint distributions of the variables are multivariate normal, then the first and second moments completely determine the distributions. On average, the imputation should give reasonable predictions for the missing data, and variability among them should reflect an appropriate degree of uncertainty. An imputation model must preserve all important associations among variables in the dataset, including interactions.

We propose a novel nonparametric multiple imputation (MI) algorithm. The remainder of this paper is organized as follows: a brief review of previous work in Section 2, details of novel algorithms in Section 3, details of how we assess the new algorithms in Section 4, results and discussion in Section 5, followed by summary and conclusions in Section 6.

2. Review of existing techniques

Missing data handling methods can be broadly classified into two categories: deletion and imputation. Listwise deletion [4] drops an entire case (subject) if a value of any explanatory variable is missing. This is a straightforward process, but cases with missing values should not simply be ignored because it will reduce sample size and will induce systematic selection bias.

Imputation methods replace missing values with values estimated from the available data. The advantage of imputation is that it uses 'expensive to collect' data, that would otherwise be discarded. Imputation techniques can be split into procedures grounded on non-model vs. model-based approaches. The end products of *non-model based approaches* are surrogate values to replace missing data. In contrast, the end products of *model based methods* are the estimated values of model parameters, which in turn are used to impute missing values.

The most commonly used *non-model based procedures* are zero imputation [4] and mean substitution [5]. In zero imputation, missing values are always replaced by a zero value. This method is simple and provides all cases with complete data. However, this method does not utilize any information about the data. The integrity and usefulness of the data can be jeopardized as a result. Mean substitution is an improvement over zero imputation. It replaces missing values of a given variable with the mean value (unconditional mean) for that variable. Its main advantage is that it produces "internally consistent" sets of results. However, this method does not take into account the relationship among variables and can distort the multivariate relationships. Mean substitution artificially reduces variance. Severe biases can result when the missing data characteristic is MAR but not MCAR.

The most sophisticated techniques for the treatment of missing values are *model-based*. These methods consider interrelations among variables. The model-based techniques can be classified into two categories: explicit model based algorithms and implicit model based algorithms. *Explicit models* create a parametric representation of the dataset. These are based on a number of assumptions. A statistical model provides accurate estimates only when model assumptions are satisfied. If the assumptions are violated, the validity of imputation values derived from applying these techniques may be in question. Commonly used explicit model based methods include expectation maximization algorithms (EM) [6] and data augmentation (DA) algorithm or Markov chain Monte Carlo (MCMC) [7].

Implicit model based algorithms often have semi-parametric or non-parametric flavours. These methods make few or no distributional assumptions about the underlying phenomenon that produced the data. Hot deck imputation methods employing best match (donor) values are the most popular implicit model based algorithm. Hot deck imputation procedures replace missing values on incomplete records using values from similar, but complete records of the same dataset. Past studies suggest that this approach is promising [5]. There are two popular variants of hot-deck imputation algorithm: *K*-nearest neighbour (KNN) imputation algorithm [8] and weighted *K*-nearest neighbour imputation algorithm (WKNN) [9]. A major limitation of these methods is the difficulty in defining what is 'similar'. Recently a number of studies applied multilayer perceptrons (MLP) [10], radial basis function networks (RBFNs) [11], and generalized regression neural networks (GRNNs) [12] to impute missing values. However, these techniques (except for GRNNs) are quite complicated, with many free parameters that can affect the quality of the imputation. Estimating all parameters simultaneously induces errors. In contrast to other neural networks, GRNN has only one free parameter, but GRNN leads to potentially severe curse of dimensionality effects [13]. The curse of dimensionality is a serious issue, especially when the missing value percentage is high.

A few studies have explored ensemble learning for missing data imputation [14]. Neural network ensemble is a machine learning paradigm where multiple neural networks (base learners) are trained to solve the same problem. The ensemble network can be either homogeneous or heterogeneous. The heterogeneous ensemble network obtains the overall output from different independent network structures, whereas the homogeneous ensemble network obtains the overall output from similar independent network structures. Methods of homogeneous ensemble model generation basically rely on varying the parameters related to the design and the training of neural network. Typically, an ensemble (homogeneous and heterogeneous ensembles) is constructed in two steps. In the first step, a large set of candidate base classifiers (neural networks) is generated and then an optimal subset of base classifiers, from the base classifiers pool, is selected to form the ensemble of classifiers. To ensure diversity in the homogeneous ensemble set, the random sub-spacing strategy is used. In the random subspace method, many different features subsets are randomly chosen for producing candidate component classifiers and then base classifiers are iteratively refined using sequential hill-climbing or stochastic techniques. On the other hand, for heterogeneous model generation, neural network ensemble members are created by using different neural network types. After the base classifiers are constructed and refined, an optimal subset of ensemble members are selected from the pool of candidate classifiers. An

effective (homogeneous and heterogeneous) ensemble should consist of high-accuracy classifiers that disagree on their predictions. Popular methods for selecting an optimal subset of ensemble members include principal component analysis (PCA) [15], correlation analysis [16], genetic algorithm (GA) [17], particle swarm optimization (PSO) [18], and hill-climbing (HC) [19]. However, these techniques are not ideal techniques for constructing base classifiers and selecting a subset of ensemble members, since they are plagued by local minima and premature convergence problems. At the second step of ensemble construction, the predictions of the learned models are integrated. The most popular combination schemes are majority and weighted voting [20]. The majority voting approach involves averaging the predictions of the individual networks. Majority voting rule constitutes a very appealing method due to its conceptual and implementational simplicity. However, this approach treats each member equally, i.e., it does not stress ensemble members that can make more contribution to the final generalization. The weighted voting approach is an improvement over the majority voting approach. In weighted voting approach, the total weight is one and each member of the ensemble is entitled to a portion of this total weight according to their performance. The ensemble decision on a new instance is obtained by a weighted vote of all ensemble members. Weights are adjusted using iterative prediction-error minimization method. The popular weight optimization methods include gradient descent algorithm [21], GA [22] and PSO [23]. A major limitation of conventional weighted voting approach is that it is a static approach. The weights for ensemble members do not depend on the instance to be classified.

Both explicit and implicit model based methods are further divided into single and multiple imputation methods. In single imputation, a single parameter is estimated with no sense of how this parameter estimate might vary across equally plausible sets of missing data imputations. Single imputation produces a single filled-in dataset, where each missing value is replaced with a single value. The replaced values are then treated as if they were actual values. In general, single imputation offers the advantage of allowing complete data analysis methods to be used, and it requires less work to impute each missing value only once. Expectation maximization (EM SI), multilayer perceptrons (MLP SI), radial basis function networks (RBFN SI) and hot deck imputation algorithms (HD SI) are examples of single imputation algorithms. A major problem with single imputation is that this approach cannot reflect sampling and imputation variability [24]. Therefore, the estimated variances of the parameters are biased toward zero, leading to statistically invalid inferences. Rubin [25] proposed multiple imputation (MI) to solve this problem . A detailed summary of MI is given in Rubin [1], Rubin and Schenker [26], Schafer [3], and Schafer and Olsen [27]. In MI, the focus is on getting the joint probability density function of model parameters that represent sampling and imputation variability. In this approach, model parameters are randomly drawn from the distribution and each missing datum is replaced by $m > 1$ possible values to accurately reflect uncertainty and to preserve data relationships and aspects of the data distribution. Multiple hot deck imputation algorithms (HD MI), MCMC, and multiple neural networks (NN MI) are examples of MI algorithms. It requires that the analyst specifies an imputation model, imputes several data sets, analyses them separately, and then combines the results. MI builds on the advantages of single imputation. MI not only allows the use of complete-data analysis methods for the data analysis, but also incorporates random error since it requires random variation in the imputation process. MI produces improved estimates of standard errors when compared with single imputation methods because repeated estimations are used. It can accommodate any model and any data and does not require specialized software. MI also increases efficiency of parameter estimates because it minimizes standard errors and simulates proper inferences from the data. The three disadvantages of MI when compared with other imputation methods are: (a) more effort to create the multiple imputations, (b) more time to run the analyses, and (c) more computer storage space for MI-created datasets [1]. These are hardly issues with current development in computer technology. However, several authors have raised questions with regard to the validity of MI approach [28,29]. They report that MI procedure tends to yield longer confidence intervals. In other words, this approach may overestimate standard error or variance (the standard error squared) of missing value estimates.

3. Novel algorithms: GEMI and GESI

We developed two novel missing value imputation algorithms—GRNN-ensemble for multiple-imputation (GEMI) and GRNN-ensemble for single imputation (GESI). Both algorithms construct an ensemble model of generalized regression neural networks (GRNN)—a well respected computational intelligence based algorithm proposed by Donald Specht in [30] (see Appendix A for a brief overview of the GRNN). The proposed algorithms employ a novel algorithm SAGA to optimize the ensemble makeup [31] (see Appendix B for a brief overview of SAGA). In GESI, one replacement value is created and imputed for each missing observation. In GEMI, several missing values are independently imputed, creating multiple (m) datasets and multiple estimates, one for each replication of the imputation process. The multiple estimates are averaged over the m imputations to create a single MI estimate of the statistic of interest. To account for the uncertainty of the imputation process, MI variance estimators that incorporate both within- and between-imputation components are used to account for imputation-related variance in the overall variance of the estimate of interest. In this section, we discuss the advantages and principal motivations behind design choices for our novel algorithms and advantages to reap from our algorithms are discussed below (the pseudocodes of GEMI and GESI are presented at the end of this section).

First, a good imputation algorithm should use as much information as possible. A variable X may be related both to the missing data mechanisms and to the actual outcomes of the covariates. To produce high-quality imputations for a particular variable$X$, the imputation model should include variables that are (a) potentially related to X and (b) potentially related to the missing mechanism of X. A general guideline is that the imputer should use a model that is general enough to preserve any associations among variables (two-, three-, or even higher-way associations) that may be the target of subsequent analysis. Hence, our algorithms attempt to use the maximum possible information content available to yield optimal performance. Our novel algorithms (GEMI and GESI) conjointly model the data and missing data mechanisms to exploit information, not only from observations, but also from presence and absence of values in other variables. This is a difficult task to accomplish. For each variable X_i, we define a corresponding indicator variable I_i, to indicate whether the value of the variable X_i is missing or not. However, adding all those indicator features comes at a price: it increases the dimensionality of the patterns. Consequently, the curse of dimensionality problem appears and the prediction performance degrades severely.

In order to ensure the inclusion of all the important predictors and interactions in the model, our algorithms follow a novel homogeneous ensemble framework that can cope with the curse of dimensionality. GEMI and GESI are GRNN ensembles where a

collection of GRNNs is trained for the same task. Ensemble members are trained on different subsets of features. These separately trained GRNNs are then combined to form one unified prediction model. There are numerous ensemble approaches available in the literature. Compared with existing ensemble learning algorithms, our proposed ensemble approach is unique in the following two aspects.

(i) *Selection of ensemble members*: The selection of an optimal feature subset and the selection of an optimal set of ensemble members are similar types of combinatorial problems. Hence, the same techniques are used for both of these problems. However, these techniques fail to guarantee an optimal solution since these methods have the shortcomings of local minima and slow convergence speed. Our novel algorithms apply our proposed feature subset selection algorithm (SAGA) to select several parsimonious feature sets with excellent prediction performances. SAGA is a hybrid algorithm based on (1) simulated annealing (SA) algorithm, (2) genetic algorithm (GA), (3) hill-climbing (HC) algorithm, and (4) GRNN (see Appendix B). These subsets are used to train separate GRNNs. GEMI and GESI then identify a parsimonious set of trained GRNNs (ensemble members) for the ensemble model using the SAGA. We demonstrated in [31] that SAGA offers the most parsimonious and best-fitting models.

(ii) *Method of combining the outputs of ensemble members*: Weighted voting is widely used for combining the outputs of several ensemble members trained independently to perform a prediction task. A weighted voting system is one in which the preferences of some voters carry more weight than the preferences of other voters. The major disadvantage of traditional weighted voting system is that the weights for each ensemble member's vote do not depend on patterns to be learned. Recent studies show that when the performance of the ensemble members is not uniform for all patterns, the efficiency of this type of voting is affected negatively [32]. To combat this, a GRNN is used as a combiner. The combiner GRNN uses the outputs of the base GRNN classifiers as inputs and produces the final prediction for the unobserved items. Fig. 1 illustrates the basic framework for the proposed GRNN ensemble. GRNN is a local approximation algorithm. To embed this feature into our own algorithms (GEMI and GESI), we used GRNN as base learners as well as the ensemble combiner. Consequently, GEMI and GESI are dynamic weighted voting methods whose combiner assigns weights to base classifiers on a pattern-by-pattern basis.

In addition to the large number of predictor variables, there is another major pitfall to be aware of when imputing missing data. Missing values are determined using complete case analysis. The sample size decreases as the percentage of missing values increases. A small sample size tends to result in a poor model fit. In order to reduce this problem, our algorithms (GEMI and GESI) imitate the iterative computation of the MCMC and EM algorithms, although these new algorithms do not use the maximum likelihood approach. GEMI and GESI repeatedly alternate between two steps, the *M* (maximization)-step and the *E* (expectation)-step. In the first *M*-step, GEMI and GESI fit the imputation model based on the complete cases only. Then, in the first *E*-step, missing values are imputed by the imputation model estimated at the previous *M*-step. Given a complete sample, the next *M*-step updates the imputation model. This new imputation model is then used in the next *E*-step for re-estimating missing values. This process continues until the algorithm converges.

Second, our proposed algorithm GEMI is a multiple imputation (MI) algorithm. MI accounts for data by restoring not only the natural variability in the missing data, but also by incorporating the uncertainty caused by estimating missing data. GEMI uses Rubin's multiple imputation framework [1] that allows us to create proper multiple imputations in complex multivariate settings. To perform multiple imputations using GEMI, we create 30 training sets since a larger number of samples will give a better estimate of the random error, every time randomly selecting 70% of the available data. Each dataset is analyzed. With 30 training sets, 30 separate sets of parameter estimates (i.e. conditional mean and conditional variance) are obtained. Individual parameter estimates are combined to get global parameters of the conditional target distribution for the missing data. We then create 30 replications of the original datasets, resulting in 30 datasets. For imputing missing values, GEMI simulates draws from the distribution of interest. The replicas of a record will differ in imputed values but not in observed values. Variation in estimates across these multiple datasets permits estimation of overall variation, including both sampling and imputation variance.

Third, both novel algorithms (GEMI and GESI) are based on a fuzzy clustering scheme. This is a non-parametric algorithm and avoids distributional assumptions. Another advantage of our methods is that they are local approximators whereas many conventional imputation algorithms such as MCMC, EM, and MLPs are global approximators. Global approximators formulate one predictive formula for the entire search space. In contrast, local approximators formulate many effective formulas in order to address local variations. GEMI and GESI have inherited these qualities from their underlying algorithm GRNN. In GRNN, all training patterns form their own clusters. When a new pattern is presented to the GRNN, each cluster assigns a membership weight (between 0 and 1) to indicate the strength of the membership of the pattern belonging to the cluster using a Gaussian membership function based on the similarity. However, it is worth noting that GRNN does not assume that data follow a Gaussian distribution. In GRNN, the outputs of new patterns are calculated from a weighted average of outputs of training patterns.

Fourth, there are only a few parameters in GEMI and GESI that need to be selected. These parameters include the parameters of GRNN and the parameters of SAGA. Apart from the above mentioned parameters, there is one extra parameter (number of imputations: m) in GEMI. GRNN has only one adjustable parameter (the smoothing parameter σ). However, we have discovered the best default value for this parameter. We set the default value for each centre's width (σ) within each GRNN to twice the mean of the distance to the 20 nearest neighbours. The width of each prototype pattern (i.e. centre) within each GRNN is different since each prototype pattern has a unique set of 20 different neighbours. As a result, the proposed default value of σ can keep the local approximation ability of GEMI and GESI intact. Table C1 shows performance impact of different σ values.

The choice of SAGA parameter values and the total number of imputations (m) depends on the computational cost one is willing to incur. Hence, the task of specifying m and SAGA parameters is relatively straightforward.

We determine default settings through experiments with 200 synthetic datasets (these 200 datasets were not used in the missing data imputation study). Users can accept default parameter values where resource constraints permit, since default parameter values will almost always produce good results. They can also set much higher values for these parameters (parameter m and parameters of SAGA). Usually, the higher parameter values provide better results. In our missing data imputation experiments, we used default settings. A reasonable number of different

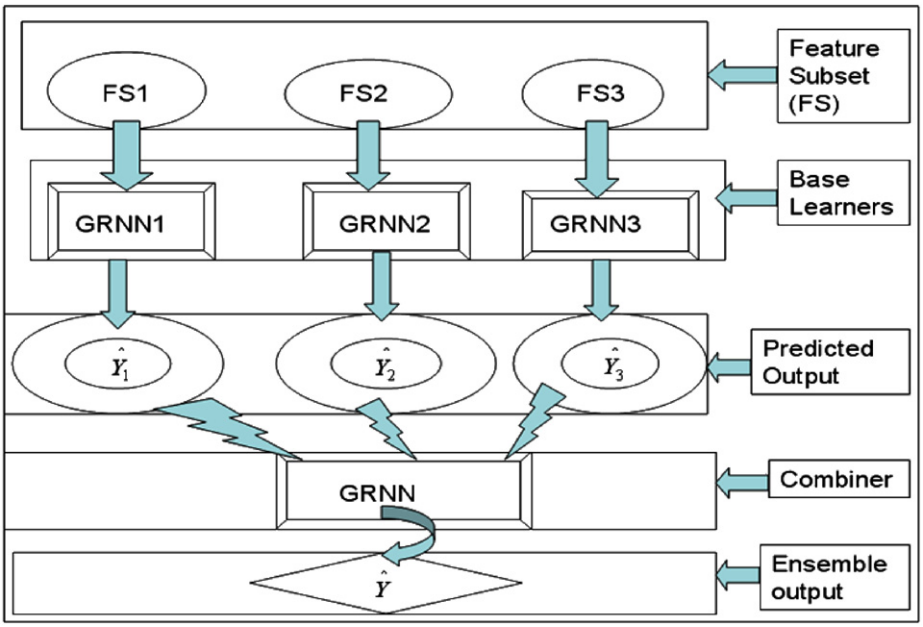

Fig. 1. The framework of the proposed ensemble classifier.

Table 1
Default parameter settings for GEMI and GESI.

Parameter	Default setting
Number of imputations m (a parameter of GEMI)	30
Smoothing factor parameter σ (for GEMI and GESI)	(each centre's width is set to average Euclidean distance to 20 nearest members)*2
Parameters of SAGA	
Population size of SA, and GA	100
Population size of HC	10
Stopping criterion of the SA and GA	Stop the algorithm if the best solution does not improve in the last 100 runs.

parameter values were tried in order to find those values for which the code seemed to perform best overall. The relative performance of different parameter values were compared using statistical tests (Friedman two-way analysis of variance by ranks and comparisons of groups or conditions with a control [33]). A brief description of these tests is available in Appendix D (see Siegel and Castellan [33] for detailed description of these statistical tests). The list of default values of parameters, stored in the configuration file in our novel algorithms, are given in Table 1.

Based on the resource constraint, we set the value of m to be 30. Ideally, the selection of the parameter value m should depend on the resource (e.g. memory and computational cost) availability, which we mentioned earlier. Our research demonstrates that precision improves increasing the number of imputations (m). Our aim is to clarify this implication, so that the analyst may make an informed choice of the parameter value. Hence, we reported the performance impacts of different m values in Table C2. It appears that the best value for parameter m is 100.

3.1. The pseudo-code of GEMI and GESI (all vectors are presented in bold face)

In the implementation of GESI, we execute only the first two steps of the full procedure outlined in this section. GESI fits a single ensemble model (model-$\boldsymbol{P}$), as shown in Fig. 2. Model-$\boldsymbol{P}$ is a

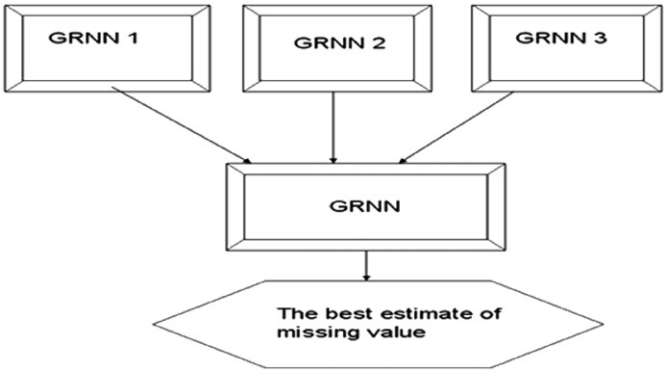

Fig. 2. Imputation model built by GESI.

set of models $(p_1,p_2,\ldots,p_n)$ in which each element $(p_1,p_2,\ldots,p_n)$ is a single ensemble model (SEM) that estimates the conditional mean of a missing value, where n is the total number of missing values.

For the implementation of GEMI, the full process has to be adhered to. GEMI develops two prototype models—$\boldsymbol{P}$ (estimates conditional means of missing values) and $\boldsymbol{Q}$ (estimates conditional variances of missing values). GEMI fits m sets of (identical in the sense that they consist of the same feature ensembles, but non-identical in the sense that they are trained on different sets of training examples) replicas of a pair of prototype ensemble models—model-$\boldsymbol{P}$: $(\boldsymbol{P_1},\boldsymbol{P_2},\ldots,\boldsymbol{P_m})$ and model-$\boldsymbol{Q}$: $(\boldsymbol{Q_1},\boldsymbol{Q_2},\ldots,\boldsymbol{Q_m})$—as shown in Fig. 3. $\boldsymbol{P_1}$, $\boldsymbol{P_2}$,…,$\boldsymbol{P_m}$ and $\boldsymbol{Q_1}$, $\boldsymbol{Q_2}$,…,$\boldsymbol{Q_m}$ are replica models.

Both prototype models ($\boldsymbol{P}$ and $\boldsymbol{Q}$) and each pair of replica models $[(P_1,Q_1),(P_2,Q_2),\ldots,(P_m,Q_m)]$ consist of n sub-models (one model for each of the n missing values). $(p_1,p_2,\ldots,p_n)$ and $(q_1,q_2,\ldots,q_n)$ represent the submodels of the prototype models P and Q, respectively. The model-$\boldsymbol{P}$: $\boldsymbol{P}=(p_1,p_2,\ldots,p_n)$ predicts the conditional means of missing values, whereas the model-$\boldsymbol{Q}$: $\boldsymbol{Q}=(q_1,q_2,\ldots,q_n)$ predicts the conditional variance of each missing value. Since the training samples of the prototype model $\boldsymbol{P}$ may contain incomplete observation vectors, the prototype model $\boldsymbol{P}$ is fitted through the application of an iterative EM-style algorithm where the M-step learns the imputation models and the E-step uses imputation models to estimate missing values. The training

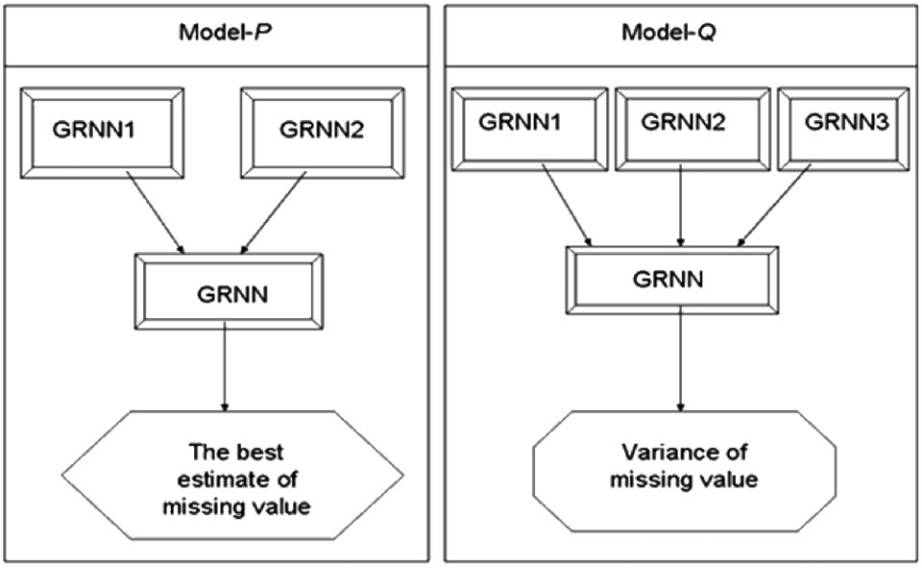

Fig. 3. Protype models built by GEMI.

samples of prototype model ***Q*** do not contain any missing data for any of the attributes. Hence, the model ***Q*** is fitted in one pass to the squared residuals of the prototype model ***P.***

Step 1: Normalize each variable to the range [0,1] using min-max normalization method. Min-max normalization subtracts the minimum value of a variable from each value of the variable and then divides the difference by the range of the attribute.

We normalize each variable before applying GEMI and GESI, i.e. make all variables to vary in the range [0,1]. This main reason for scaling the data to this range is to prevent the inputs with much larger ranges from dominating in the training. Another useful, but more complicated, pre-processing strategy would be to whiten the observed variables.

Step 2: *Design and construction of prototype model-P*: As mentioned before, in Model-***P***, there is a set of models (p_1,p_2,...,p_n) in which each element (p_1,p_2,…,p_n)is a single "ensemble" model (SEM_m) that estimates the conditional mean of a missing value, where n is the total number of missing values. In other words, in Model-*P*, there is a SEM_m for each missing value in the dataset, wherein the variable with the missing value is the target variable. We define an indicator variable I for each main variable X with $[I_{ij}]=1$ if X_{ij} is missing and 0 otherwise. Here, X_{ij} denotes the value of the jth main variable in the ith pattern. I_{ij} denotes the value of the jth indicator variable in the ith pattern. The main ($X=[X_1,X_2...]$) and indicator ($I=[I_1,I_2,...]$) variables are considered as candidate input variables.

We note that there can be multiple missing values in a record and our algorithms can deal with this issue. The proposed imputation algorithm is an iterative algorithm. Each iteration consists of two steps: *M* step and *E* step. In *M* step our algorithm fits multiple models (one model for each of the missing values). These models can handle only one missing variable for each case. In the first iteration, each of these models includes only those independent variables that have no missing values in the test pattern. Listwise deletion is performed to remove training cases with missing values in one or more independent variables. Missing values are computed in the *E* step of the first iteration of GEMI and GESI using the models fitted in the first *M* step. Thus, the first *E* step will create a new complete matrix or dataset. On the next iterations, no deletion is required. The *M*-step of each iteration fits imputation models for missing values of the original dataset to the data imputed at the *E*-step of the last iteration. Then the new imputation models are applied to re-compute the missing values of the original matrix in the next *E*-step, thereby forming a new matrix which serves as the training sample for the next *M*-step. SAGA is used in each iteration to select an optimal subset of features. It is worth noting that the observed values remain unchanged, only missing values are re-estimated in each iteration. The iterations continue until the estimates of the missing values become stable.

The pseudo-code of iterative *M* and *E* steps loop is given below.

Step 2.1: *The maximization M-step*

Create a loop to build a SEM_m model for each missing value

$k=1$; $n=$total number of missing values

While $k \leq n$

Step 2.1.1: Use SAGA to identify 100 (or less, when the total number of diverse solutions found so far is less than 100) of the best feature subsets (FS) for the target variable from the pool of candidate input variables (both the main and indicator variables) using 10 fold cross validation. The parameters of SAGA are fixed at the values shown in Table 1.

Step 2.1.2: Train a separate GRNN with each of these 100 feature subsets. This will give us 100 trained GRNNs. These GRNNs are trained to predict the output variable (i.e. the variable with missing values).

Step 2.1.3: Use SAGA to select an optimal subset of base GRNN classifiers from the pool of 100 trained GRNNs to form the ensemble model (p_k) using 10-fold cross validation. The parameters of SAGA are fixed at the values shown in Table 1.

Step 2.1.4: Complete the construction of the ensemble model (p_k) by training the combiner GRNN to predict the conditional mean of the missing value d_{ij} using the outputs of the base learners as inputs.

$k=k+1$;

End (while $k \leq n$ loop)

Step 2.2: *The expectation E-step*: *E*-step supplies the imputed values based on imputation models calculated in the previous *M*-step.

Given a complete sample, the next *M*-step updates the model $P=(p_1,p_2,...,p_n)$ and then the next *E*-step re-estimates the missing values based on the updated models. The two steps are iterated until the conditional means of missing values become stable. The process of developing model *P* is further illustrated in Fig. 4.

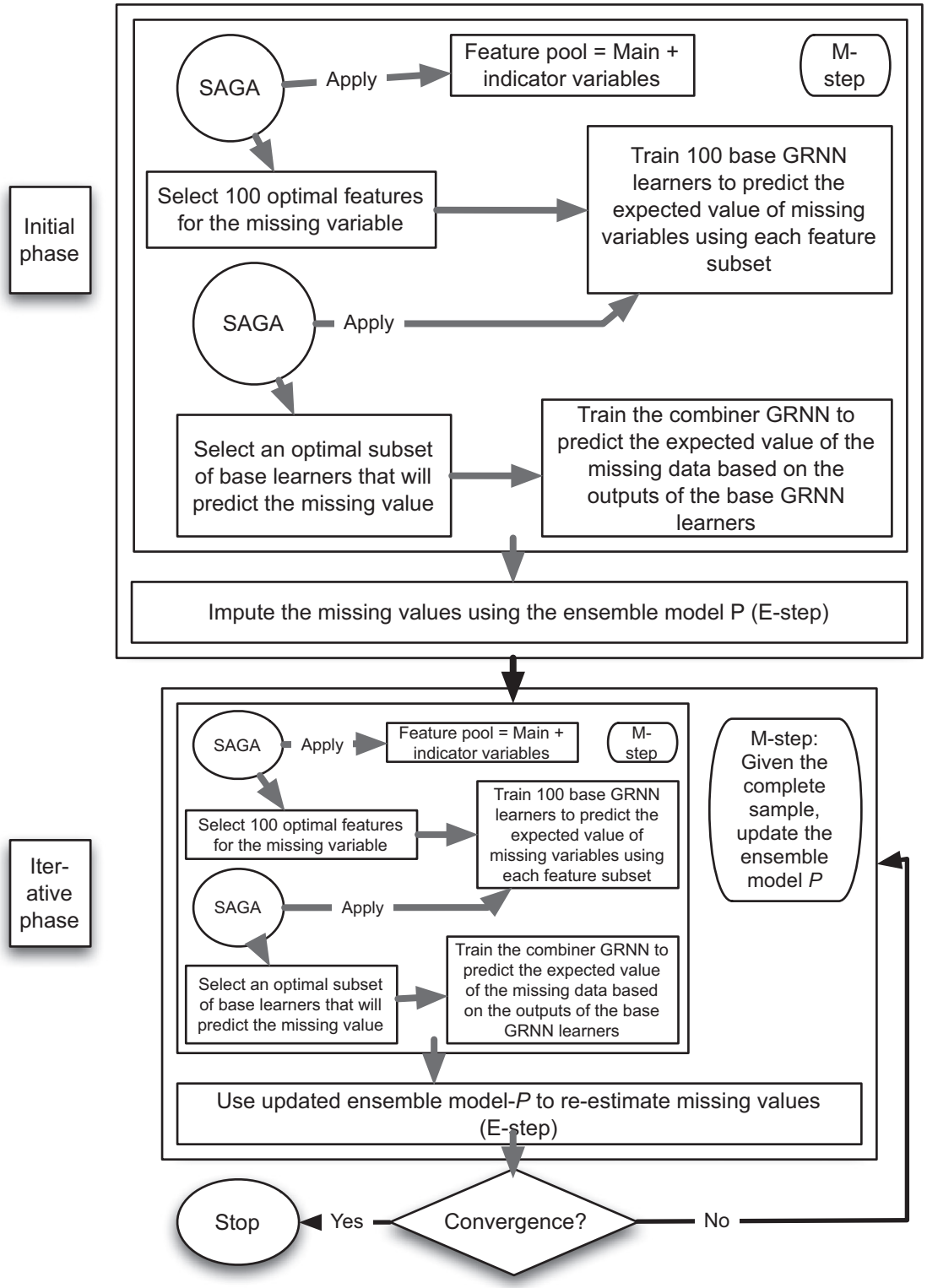

Fig. 4. A block diagram of the construction of prototype model-*P* for the conditional mean of missing data.

Step 3: *Design and construction of prototype model-Q*: Model-**Q** is a set of models $Q=(q_1,q_2,...,q_n)$ in which each element $(q_1,q_2,\ldots,q_n)$ is a single "ensemble" model (SEM_v) that estimates the conditional variance of a missing value. In Models-Q, there is a SEM_v for each missing value in the dataset. In other words, there is a SEM_v model associated to each SEM_m model. To construct a SEM_v model, the following steps were executed.

$K=1$;
$n=$total number of missing values
While $K \leq n$
// build the model q_k

Step 3.1: *Define input and output variables for the model* (q_k): The target variable of the model q_k is defined as the squared residuals of the model p_k on the training set. All variables, except

the variable with missing values (i.e. the target variable of model p_k), and the corresponding indicator variables are treated as candidate input variable for the model q_k.

Step 3.2: *Apply SAGA to select the ensemble of base learners*:

Step 3.2.1: Identify 100 near-optimal but diverse feature subset solutions using 10-fold cross validation. The default parameter setting used in the experiment is listed in Table 1.

Step 3.2.2: Train a base GRNN learner using each of the 100 feature subset solutions.

Step 3.2.3: Apply SAGA to select an optimal subset of GRNNs from the pool of 100 trained GRNNs, which will be used to form the model q_k. Table 1 summarizes the SAGA parameter setting for the experiments.

Step 3.2.4: To complete the construction of the ensemble model q_k, train the combiner GRNN to predict the expected squared error of the ensemble model p_k based on the outputs of the base GRNN learners.

$K=K+1$;

End (while $K \le n$) loop

The model-Q construction process is illustrated in Fig. 5.

Step 4: *Construct multiple models for estimating parameters of missing observations*: Generate m exact replicas ($(\boldsymbol{P_1},\boldsymbol{Q_1})$,$(\boldsymbol{P_2},\boldsymbol{Q_2})$,…, $(\boldsymbol{P_m},\boldsymbol{Q_m})$) of the prototype models $\boldsymbol{P}$ and $\boldsymbol{Q}$. Train the replicas using 70% training data rather than 100%. The training set of each replica was selected randomly. While training the different replicas, we did not apply SAGA for selecting ensemble members since these ensemble models are constructed from the prototype configuration. The replicas were trained using the following steps:

//Use a loop to train replica models

$K=1$;

While $K \le m$

Step 4.1: Train each member GRNN of the replica ensemble model P_k on the training set using input (main and indicator variables) and output (variable with the missing value) data.

Step 4.2: Present the training patterns to model P_k for the prediction of target variable and compute squared residuals.

Step 4.3: Train each member GRNN of the replica model Q_k using inputs (main and indicator predictor variables) and output (squared residuals of P_k) data.

$K=K+1$;

End Loop/*Replica models ($(\boldsymbol{P_1},\boldsymbol{Q_1})$,$(\boldsymbol{P_2},\boldsymbol{Q_2})$,…,$(\boldsymbol{P_m},\boldsymbol{Q_m})$) are now ready for use.*/

The process of developing a pair of replicas of models P and Q is summarized in Fig. 6.

Step 5: *Estimate the parameters of missing data*

$L=1$;

n=number of missing values

m=number of replica models

While $L \le n$

$k=1$;

While $k \le m$

- Present the fully trained model $\boldsymbol{P_k}$ with the missing value d_L for predicting the conditional mean ($\widehat{Y}_{Lk}$) of missing value (d_L).
- Present the missing value (d_L) to the trained model $\boldsymbol{Q_k}$ for predicting the conditional variance ($\widehat{U}_{Lk}$) of missing value (d_L).
- End While loop $k \le m$
- The conditional mean of the missing value d_L is the average of the single estimates:

$$\widehat{Y}_L = \frac{1}{m}\sum_{k=1}^{m} \widehat{Y}_{Lk} \tag{1}$$

- Estimate the within-imputation variance (i.e. sampling error)

$$\overline{U}_L = \frac{1}{m}\sum_{k=1}^{m} \widehat{U}_{Lk} \tag{2}$$

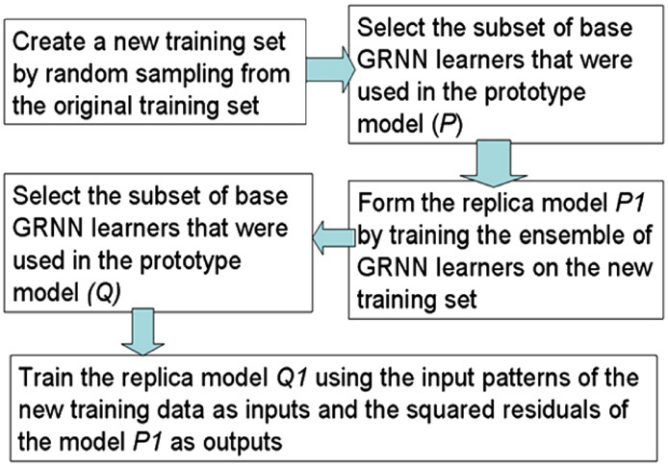

Fig. 6. Generating a pair of replica ensemble models.

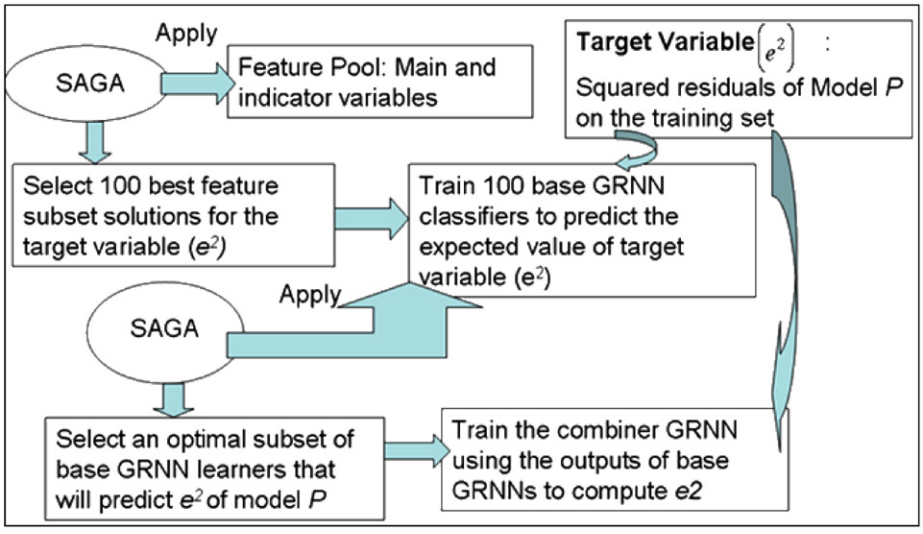

Fig. 5. A block diagram of the construction of the ensemble model Q for the conditional error variance of missing data.

- Estimate the between-imputation variance (i.e. the imputation error)

$$B_L = \frac{1}{m-1}\sum_{k=1}^{m}(\widehat{Y}_{Lk}-\overline{Y}_L)^2 \quad (3)$$

- Next, we compute the total variance of the missing value (d_L). Typically, total variance equals *within-variance* ($\overline{U}_L$) plus *between-variance* (B_L). However, Rubin's approach is to weight the between-imputation variance according to the number of imputations performed. Thus the total variance (T) is computed by the following formula:

$$T_L = \overline{U}_L + \left(1+\frac{1}{m}\right)B_L \quad (4)$$

//We now know the mean and variance of the missing value d_L.
$L=L+1$;
End (of While Loop $L \le n$)
//We now know the conditional means and variances of all missing values.

Step 6: *Replace each missing value by m plausible values: Replicate each record of the original dataset m times.*

We then impute the missing values setting them to $\overline{Y}_L + \sqrt{T_L}R$ where R is a pseudo-random number drawn from a normal distribution with mean 0 and standard deviation 1.

Steps 5 and 6 are further illustrated graphically in Fig. 7.

4. Comparative performance analysis

We compare our algorithms (GEMI and GESI) with popular single imputation (SI) and multiple imputation (MI) algorithms: (1) expectation maximization (EM) SI, (2) EM MI, (3) generalized regression neural networks (GRNN) SI, (4) GRNN MI, (5) hot-deck (HD) SI, (6) HD MI, (7) heterogeneous ensemble of GRNNs with simple averaging (HES) SI, (8) heterogeneous ensemble of GRNNs with simple averaging (HES) MI, (9) heterogeneous ensemble of GRNNs with simple averaging (HEW) SI, (10) heterogeneous ensemble of GRNNs with simple averaging (HEW) MI, (11) homogeneous ensemble of GRNNs with simple averaging (HOS) SI, (12) homogeneous ensemble of GRNNs with simple averaging (HOS) MI, (13) homogeneous ensemble of GRNNs with weighted averaging (HOW) SI, (14) homogeneous ensemble of GRNNs with weighted averaging (HOW) MI, (15) K-nearest neighbour algorithm (KNN) SI, (16) KNN MI, (17) Markov chain Monte Carlo (MCMC), (18) multilayer perceptrons (MLP) SI, (19) MLP MI, (20) mean substitution (MS), (21) radial basis function neural networks (RBFNN) SI, (22) RBFN MI, (23) weighted KNN (WKNN) SI, (24) WKNN MI, and (25) zero imputation (ZI) on 98 datasets (30 synthetic datasets+67public real-world datasets+1 new real-world dataset), using 10 fold cross validation. The public real-world datasets are obtained from UCI machine learning repository [34]. The implementation procedures of the conventional algorithms are briefly described in Appendix E.

Appendix F presents a brief description of datasets. Synthetic datasets and public real-world datasets are complete with no missing data. These datasets were used to conduct controlled experiments. We artificially removed data using ignorable and non-ignorable missing value mechanisms at different rates of missing values. Appendix G details the methods adopted to simulate random and non-random missing data mechanisms. Roughly 33% of the total missing values were introduced, using the MCAR mechanism and approximately 33.5% using MAR mechanism into each of the datasets. The remaining missing values were imposed on the datasets using the MNAR missing value mechanism. Then the imputation algorithms were applied separately to each incomplete dataset for imputation of missing values. We compare performance of algorithms on synthetic and public real-world datasets in three different ways:

First, each multiple imputation algorithm's performance was evaluated based on how accurately the algorithm recovers the nature of the conditional posterior distribution of the missing value. Each multiple imputation algorithm constructs a 95%

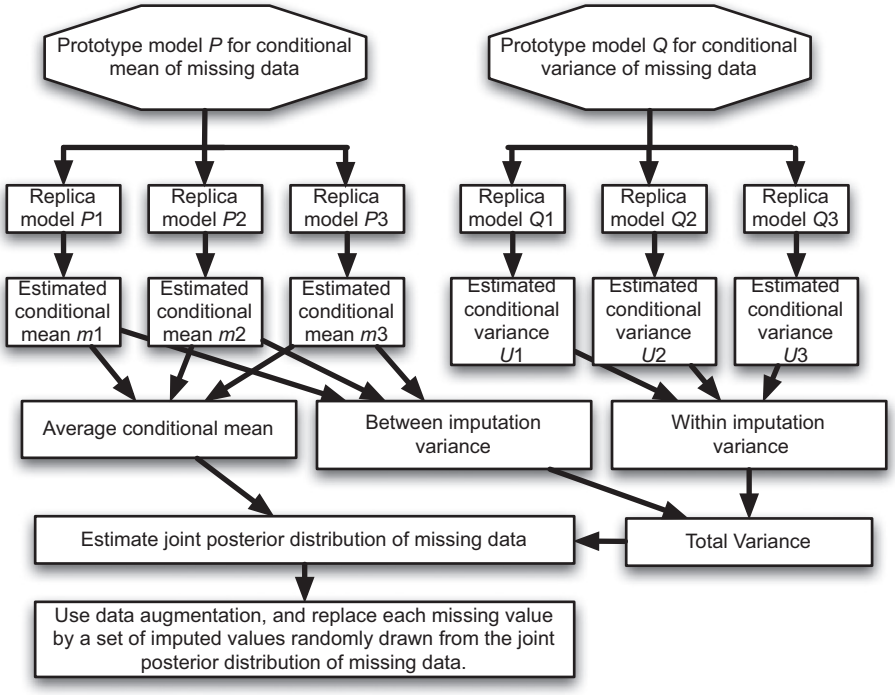

Fig. 7. Multiple imputation procedure implemented in GEMI.

confidence interval estimate of the missing value. The formula for the 95% confidence interval using the normal approximation is: (conditional mean $\pm 1.96\sqrt{\text{conditional variance}}$).

We examine the relative performance in the sense of interval estimation accuracy of multiple imputation algorithms. In general, an interval forecast is considered to be correct if the actual value falls inside the predicted 95% confidence interval. The higher the accuracy of interval identification, the better the algorithm. If the two algorithms have the same interval estimation accuracy on a dataset, the algorithms are ranked based on the average length of the estimated confidence interval. A narrow confidence interval implies high precision. For example, if we let the confidence interval be $(-\infty,\infty)$, then we may achieve 100% accuracy, but this interval is associated with very poor precision. Single imputation algorithms were naturally excluded from this study since they only provide a point estimate of the missing value.

Second, we compare the imputation algorithms with respect to the precision in terms of the accuracy of the missing value estimates using 10-fold cross validations on datasets. In general, the higher the accuracy values, the higher the agreement between observed and predicted data. If the variable of interest is continuous, then mean absolute percentage error (MAPE) was used to estimate the predictive accuracy of the algorithms.

$$\textit{Accuracy}\ (\%) = 100 - \textit{MAPE}\ (\%) = 100 - \frac{100}{N}\sum_{i=1}^{N}\frac{\left|X_i - \widehat{X_i}\right|}{X_i} \tag{5}$$

where N is the number of test patterns, X_i is the observed value of the variable of interest (X) at the ith point and $\widehat{X_i}$ is the predicted value.

Third, missing data inevitably affect a classifier's performance. Hence, the relative merits of imputation algorithms were assessed by measuring the impact of imputation on the classification of the response variable.

The Friedman test is used to test the null hypothesis that the performance is the same for all algorithms. After applying the Friedman test and noting it is significant, "Comparison of Groups or Conditions with a Control" tests (details are available in [33, p. 181]) were performed in order to test the (null) hypothesis that there is no significant difference between any pair of the imputation algorithms under study. Appendix C provides a brief overview of these tests.

In addition to experiments on public real world datasets, the performance was tested with a new real world dataset "Smoking Dataset". This dataset contains about 37% missing values. We did not artificially remove values from this dataset. We assess each imputation algorithm's performance on this dataset in terms of accuracy of output estimation by the GRNN classifier trained with the imputed dataset. Since all the missing values in this dataset are actually unknown to us, it was not possible to compare imputation algorithms in terms of the estimation error of missing values. Appendix F provides a brief description of the dataset.

We adopted a set of measures for free and fair competition between imputation algorithms, which we discussed below.

4.1. Measures for fair comparative evaluations of imputation algorithms

The following measures were taken to organize fair competition among all imputation algorithms:

- We determine the performance of every imputation algorithm based on the average predictive accuracy of a GRNN classifier, a MLP classifier, and a logistic regression technique in the imputed dataset using 10-fold cross-validation.
- The input space of all imputation algorithms includes main variables and corresponding indicator variables.
- All algorithms use our proposed feature subset selection algorithm SAGA for selecting an optimal subset of features. GEMI, GESI, HOS SI, HOS MI, HOW SI and HOW MI employ SAGA for the selection of optimal subsets of features, as well as an optimal strategy for selecting ensemble members. All imputation algorithms use SAGA with the parameter settings reported in Table 1.
- Although none of these algorithms (except EM and MCMC) use maximum likelihood (ML) estimation of model parameters, all algorithms (except MS, ZI, HD, KNN, and WKNN) were implemented as an (EM-style) iterative method to refine imputation models.
- Each multiple imputation (MI) algorithm replaces missing values with a set of 30 plausible values.

5. Results and discussion

Fig. 8 presents the impact of imputation of missing values by different imputation algorithms on classification accuracy for smoking dataset.

Summary results on the remaining 97 datasets (synthetic and UCI datasets) are presented in Tables 2–5. These tables compare our novel imputation algorithms with well-known imputation procedures in terms of (i) the overall mean accuracy of classifying the response variable on imputed datasets with different percentage of missing data (Table 2), (while Table 2 displays the average classification accuracy of three classifiers (GRNN, LR, and MLP), the individual accuracies of GRNN, LR and MLP in imputed datasets are presented in Tables H1–H3), (ii) the overall mean accuracy of interval estimation of the missing value (Table 3) and

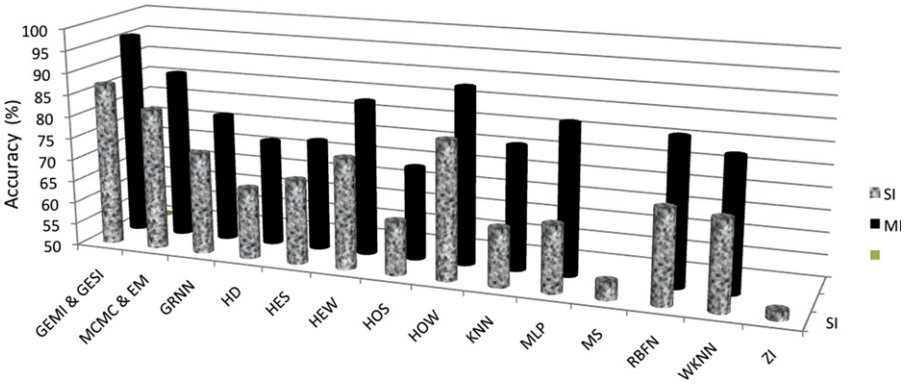

Fig. 8. Impact of imputation on classification accuracy on smoking dataset.

Table 2
Average output classification accuracy of GRNN, LR and MLP classifiers in imputed datasets under different levels of missing rates (standard deviations in parentheses).

Algorithm	Rate of missing values								
	5%	10%	20%	30%	40%	50%	60%	70%	75%
GEMI	97 (4)	**96 (4)**	**92 (6)**	**85 (7)**	**76 (9)**	**69 (17)**	**59 (17)**	**55 (12)**	44 (15)
GESI	97 (4)	85 (6)	**75 (8)**	62 (15)	55 (17)	55 (10)	51 (10)	53 (11)	40 (15)
EM	97 (4)	75 (10)	61 (13)	60 (12)	52 (20)	53 (16)	49 (9)	43 (13)	43 (13)
GRNN MI	97 (4)	79 (10)	71 (11)	68 (10)	51 (19)	52 (14)	50 (11)	44 (14)	43 (16)
GRNN SI	97 (4)	80 (11)	64 (13)	57 (19)	55 (17)	50 (9)	50 (9)	39 (14)	46 (15)
HD MI	97 (4)	81 (7)	62 (13)	52 (20)	49 (16)	54 (11)	50 (8)	39 (18)	43 (17)
HD SI	97 (4)	80 (8)	61 (8)	51 (20)	50 (15)	51 (12)	47 (8)	39 (17)	46 (18)
HES MI	97 (4)	81 (9)	71 (11)	59 (14)	49 (23)	**59 (17)**	50 (10)	39 (19)	38 (17)
HES SI	97 (4)	79 (10)	62 (15)	54 (15)	48 (20)	50 (18)	52 (8)	43 (15)	43 (13)
HEW MI	98 (5)	82 (7)	73 (8)	**70 (13)**	61 (17)	50 (10)	49 (12)	41 (16)	41 (14)
HEW SI	97 (5)	81 (9)	68 (12)	59 (12)	52 (16)	49 (9)	50 (10)	42 (17)	42 (18)
HOS MI	96 (4)	79 (10)	68 (17)	66 (19)	52 (17)	51 (16)	52 (9)	44 (15)	39 (17)
HOS SI	97 (4)	73 (15)	53 (16)	52 (18)	57 (18)	52 (13)	51 (10)	40 (19)	47 (15)
HOW MI	97 (5)	**86 (8)**	**80 (9)**	**75 (12)**	**67 (13)**	**58 (12)**	50 (11)	43 (15)	41 (17)
HOW SI	96 (4)	82 (8)	71 (14)	63 (19)	43 (16)	47 (15)	47 (9)	40 (18)	46 (15)
KNN MI	97 (4)	79 (11)	61 (13)	54 (19)	56 (16)	53 (14)	48 (11)	40 (15)	39 (16)
KNN SI	97 (4)	73 (15)	56 (18)	51 (22)	55 (18)	48 (10)	48 (8)	44 (15)	43 (15)
MCMC	97 (4)	**87 (7)**	**80 (7)**	**73 (11)**	**62 (15)**	48 (16)	50 (9)	39 (16)	42 (16)
MLP MI	96 (3)	79 (9)	65 (11)	63 (15)	57 (16)	51 (12)	52 (10)	43 (17)	41 (16)
MLP SI	97 (5)	74 (11)	56 (11)	51 (18)	54 (18)	48 (12)	42 (14)	44 (12)	42 (14)
MS	98 (4)	73 (9)	54 (11)	51 (22)	50 (21)	52 (9)	49 (11)	43 (14)	41 (15)
RBF MI	97 (4)	79 (12)	69 (12)	64 (11)	45 (20)	49 (13)	48 (11)	40 (14)	39 (19)
RBF SI	97 (4)	78 (11)	60 (11)	49 (17)	51 (13)	48 (18)	51 (8)	38 (15)	43 (17)
WKNN MI	98 (4)	78 (11)	66 (11)	60 (21)	50 (20)	51 (12)	50 (9)	43 (15)	41 (17)
WKNN SI	97 (4)	75 (13)	59 (13)	52 (22)	55 (18)	51 (12)	52 (9)	46 (16)	39 (17)
ZI	96 (5)	69 (15)	50 (18)	48 (12)	55 (15)	46 (9)	48 (8)	41 (17)	48 (15)

Table 3
Interval estimation accuracy of imputation algorithms under different levels of missing rates (standard deviations in parentheses).

Algorithm	Rate of missing values								
	5%	10%	20%	30%	40%	50%	60%	70%	75%
GEMI	94 (5)	**97 (2)**	**96 (4)**	**90 (9)**	**82 (11)**	**79 (14)**	**71 (15)**	**64 (16)**	49 (15)
GRNN MI	94 (5)	**90 (5)**	79 (6)	73 (11)	**74 (14)**	54 (17)	48 (9)	49 (12)	50 (12)
HD MI	93 (4)	**88 (4)**	70 (10)	70 (11)	**70 (10)**	55 (14)	**55 (13)**	50 (13)	51 (14)
HES MI	94 (6)	88 (6)	73 (7)	74 (7)	58 (29)	**59 (24)**	**54 (10)**	52 (13)	47 (10)
HEW MI	94 (5)	**91 (3)**	**80 (11)**	**80 (8)**	**72 (18)**	56 (13)	**54 (10)**	51 (14)	50 (11)
HOS MI	93 (5)	83 (7)	67 (6)	68 (17)	60 (11)	52 (7)	50 (8)	47 (12)	50 (14)
HOW MI	94 (5)	**90 (3)**	**87 (5)**	**80 (13)**	**74 (14)**	**62 (10)**	**56 (9)**	50 (15)	48 (11)
KNN MI	95 (5)	80 (8)	71 (9)	66 (13)	59 (18)	56 (13)	**55 (10)**	51 (13)	52 (14)
MCMC	94 (5)	**92 (2)**	**90 (5)**	**82 (12)**	**72 (17)**	54 (12)	52 (10)	49 (14)	53 (11)
MLP MI	93 (4)	83 (7)	70 (11)	67 (15)	61 (16)	50 (14)	48 (8)	51 (13)	49 (13)
RBF MI	94 (5)	85 (7)	73 (7)	74 (17)	56 (18)	**59 (22)**	50 (18)	52 (13)	50 (13)
WKNN MI	93 (5)	**89 (6)**	71 (9)	61 (19)	67 (21)	50 (7)	45 (11)	47 (13)	51 (10)

average length of constructed 95% confidence intervals for missing values (Table 4), and (iii) the overall mean accuracy of point estimation of the missing value (Table 5).

The Friedman tests reveal significant differences ($p < 0.05$) in the performance of imputation algorithms at 10–70% missing data. The classification performance of imputation algorithms based on pair-wise tests is presented in Table 6.

The pair-wise test results for other performance measures are shown in Tables H4 and H5. Table H6 compares the computational cost of imputation algorithms. All algorithms were run on a 3.40 GHz Intel® Pentium® D CPU with 2 GB RAM.

Our results lead to the following insights about the imputation algorithms:

- The rates of missing values affect the performance of the imputation algorithms (Tables 2–6 and Tables H4 and H5). There was no significant difference in the performance of algorithms when the percentage of missing values is either very low (not more than 5%) or very high (above 75%). Thus, it would appear that a difference in the performance of imputation algorithm develops when the percentage of missing values is not too high or too low.
- Our results reveal that GRNN, LR, and MLP classifiers have the highest mean accuracy across all levels of missing data when classifiers are trained on the dataset imputed by GEMI (Fig. 8, Tables 2 and 6 and Tables H1–H3). GEMI offers the best performance when the percentage of missing data is between about 10% and 70%. Within this range of missing values, GEMI outperformed all imputation algorithms in terms of the two criteria mentioned earlier: (i) the accuracy of output classification (Tables 2 and 6) and (ii) the interval estimation accuracy of missing data (Tables 3, 4 and Table H4). Imputation algorithms reach at their personal best when imputed datasets are assessed on the basis of the accuracy of the GRNN classifier (Tables H1–H3).
- In terms of the third criterion which relates to the point estimation accuracy of the missing value estimates, GESI has

Table 4
The overall width of interval estimates of missing values under different levels of missing rates (standard deviations in parentheses).

Algorithm	Rate of missing values								
	5%	10%	20%	30%	40%	50%	60%	70%	75%
GEMI	9 (7)	**16 (11)**	**23 (12)**	**39 (14)**	51 (22)	67 (18)	46 (14)	53 (21)	31 (28)
GRNN MI	8 (8)	25 (9)	35 (14)	**36 (16)**	53 (32)	60 (39)	38 (20)	32 (27)	30 (28)
HD MI	12 (9)	35 (14)	32 (12)	**28 (14)**	**28 (12)**	60 (23)	**35 (27)**	**23 (27)**	29 (24)
HES MI	10 (8)	30 (9)	36 (20)	44 (24)	**32 (25)**	64 (39)	48 (29)	33 (30)	29 (27)
HEW MI	10 (8)	29 (10)	**28 (9)**	49 (28)	82 (36)	66 (22)	55 (34)	28 (29)	30 (26)
HOS MI	9 (9)	27 (15)	47 (24)	57 (32)	60 (19)	**53 (19)**	40 (30)	24 (30)	32 (25)
HOW MI	9 (8)	**20 (8)**	**27 (14)**	49 (26)	64 (30)	67 (33)	**37 (26)**	28 (25)	34 (28)
KNN MI	11 (9)	24 (13)	35 (27)	37 (24)	40 (25)	53 (31)	**37 (30)**	31 (32)	33 (28)
MCMC	12 (9)	23 (11)	**25 (14)**	46 (26)	60 (34)	64 (30)	**36 (28)**	23 (23)	21 (25)
MLP MI	8 (7)	22 (15)	48 (25)	46 (30)	**35 (26)**	**48 (30)**	42 (28)	35 (29)	29 (29)
RBF MI	12 (10)	25 (11)	35 (23)	37 (23)	47 (29)	77 (36)	42 (30)	29 (28)	29 (28)
WKNN MI	11 (9)	22 (13)	31 (14)	**31 (20)**	44 (26)	59 (29)	41 (26)	30 (26)	41 (30)

Table 5
The accuracy of estimating the missing values under different levels of missing rates (standard deviations in parentheses).

Algorithm	Rate of missing values								
	5%	10%	20%	30%	40%	50%	60%	70%	75%
GEMI	89 (11)	81 (10)	79 (14)	**76 (12)**	70 (15)	62 (14)	62 (14)	**55 (13)**	50 (28)
GESI	89 (10)	**98 (2)**	**94 (5)**	**89 (8)**	**79 (12)**	**74 (14)**	**64 (17)**	**60 (17)**	44 (31)
EM	89 (11)	**88 (9)**	**84 (9)**	**77 (10)**	**72 (12)**	**64 (7)**	**54 (8)**	41 (30)	59 (31)
GRNN MI	91 (8)	73 (8)	64 (8)	63 (10)	57 (17)	50 (11)	53 (12)	46 (34)	53 (32)
GRNN SI	88 (11)	**89 (9)**	**89 (9)**	**77 (11)**	70 (10)	60 (13)	57 (12)	50 (29)	51 (30)
HD MI	89 (10)	66 (19)	58 (13)	61 (11)	54 (12)	50 (6)	43 (10)	53 (29)	43 (32)
HD SI	89 (12)	**90 (9)**	86 (10)	72 (11)	58 (14)	50 (7)	54 (15)	55 (32)	48 (29)
HES MI	86 (11)	68 (14)	63 (9)	61 (16)	62 (16)	51 (21)	44 (22)	45 (29)	50 (23)
HES SI	89 (9)	80 (16)	69 (13)	74 (15)	55 (22)	58 (13)	52 (12)	53 (31)	47 (34)
HEW MI	89 (11)	75 (11)	66 (13)	64 (13)	57 (11)	54 (17)	52 (12)	42 (28)	50 (34)
HEW SI	88 (11)	**88 (6)**	**85 (9)**	**78 (13)**	68 (11)	63 (11)	58 (13)	50 (35)	57 (30)
HOS MI	90 (11)	70 (13)	60 (14)	55 (14)	57 (13)	53 (8)	49 (13)	56 (33)	45 (29)
HOS SI	89 (8)	81 (11)	**81 (12)**	72 (14)	66 (18)	55 (15)	56 (13)	49 (28)	46 (33)
HOW MI	91 (11)	77 (8)	71 (17)	64 (16)	64 (12)	55 (17)	51 (13)	46 (32)	57 (34)
HOW SI	89 (10)	**89 (9)**	**85 (10)**	72 (11)	69 (15)	57 (13)	58 (13)	45 (29)	48 (35)
KNN MI	90 (10)	73 (11)	65 (9)	64 (15)	56 (18)	41 (17)	44 (17)	49 (33)	53 (31)
KNN SI	92 (10)	87 (7)	**81 (13)**	**80 (13)**	62 (14)	55 (18)	45 (15)	53 (29)	45 (32)
MCMC	86 (10)	77 (13)	70 (14)	64 (15)	58 (14)	55 (15)	53 (19)	56 (31)	51 (33)
MLP MI	88 (10)	74 (12)	61 (12)	56 (14)	50 (18)	56 (19)	53 (14)	50 (33)	48 (29)
MLP SI	86 (10)	86 (7)	**80 (9)**	**77 (16)**	68 (11)	60 (9)	58 (8)	41 (25)	40 (28)
MS	90 (11)	84 (13)	**80 (13)**	**75 (18)**	63 (18)	58 (21)	50 (13)	47 (30)	50 (33)
RBF MI	88 (9)	73 (13)	66 (12)	66 (18)	60 (14)	46 (16)	47 (18)	46 (29)	55 (23)
RBF SI	87 (13)	83 (10)	**88 (11)**	**75 (17)**	**74 (13)**	**62 (11)**	**62 (14)**	51 (33)	57 (31)
WKNN MI	87 (10)	75 (14)	64 (10)	67 (12)	54 (14)	48 (12)	47 (15)	57 (30)	50 (30)
WKNN SI	89 (9)	**88 (7)**	85 (8)	73 (12)	69 (9)	61 (10)	50 (13)	57 (29)	47 (34)
ZI	85 (11)	79 (15)	67 (17)	56 (22)	53 (21)	54 (19)	57 (19)	52 (29)	53 (30)

significantly outperformed all imputation algorithms (Tables 5 and H5). In terms of classification accuracy, GESI significantly outperformed all single imputation algorithms (Fig. 8 and Tables 2, 6).

- The "high-level" primary goals of data mining are output prediction and clustering. It is important to bear in mind that the fundamental goal of missing data imputation is to improve and facilitate the practice of these tasks. Imputation is generally not undertaken for its own sake. Both single imputation (SI) and multiple imputation (MI) approaches have been subject to criticisms. According to the critics, SI substantially underestimates uncertainty about the missing value, whereas MI overestimates the variance. However, our empirical results suggest that the performance of imputation algorithms in terms of their classification accuracy is relatively better for the MI than for the SI (Fig. 8 and Tables 2, 6). For instance, the performance of GEMI is better than that of GESI. Similarly, MCMC, HOW MI, HOS MI, HEW MI, HES MI, GRNN MI, RBF MI , HD MI, KNN MI, WKNN MI, MLP MI, are generally better than EM SI, HOW SI, HOS SI, HEW SI, HES SI, GRNN SI, RBF SI, HD SI, KNN SI, WKNN SI, and MLP SI, respectively.
- In terms of classification accuracy of the response variable on the imputed data, GEMI, MCMC and HOW MI are the top three imputation algorithms that achieved the first two places across the varying levels of missing data (Tables 2 and 6). Similarly, in the sense of interval estimation accuracy of the missing data, GEMI, MCMC, and HOW MI are the top imputation algorithms (Tables 3, 4 and H4). In the sense of point estimation of the missing data (i.e. the accuracy of the missing value estimates), the top imputation algorithms include GESI, HOW SI, HD SI, GRNN SI, EM, WKNN SI, HEW SI, and RBF SI (Tables 5 and H5). These results indicate a direct relation between the classification accuracy of the response variable and the interval estimation accuracy of the missing value estimates, since in both cases the best algorithms are the same. The results also demonstrate that the point estimation accuracy of the missing

Table 6
Pairwise comparisons among imputation algorithms in terms of output classification accuracy.

Rank	Algorithm	Significantly outperformed algorithms
With about 10% missing values		
1	GEMI	(1)MCMC, (2)HOW MI, (3)GESI, (4)HOW SI, (5)HEW MI , (6)HES SI, (7)GRNN MI, (8) HD MI, (9)WKNN_MI, (10)HOS SI, (11)HES MI, (12)HEW SI, (13)RBF MI, (14) HD SI, (15)KNN MI, (16) RBF SI, (17) EM, (18) WKNN SI, (19) GRNN SI, (20) MLP MI, (21) MLP SI, (22) HOS MI, (23) KNN SI, (24) MS, (25) ZI
2	MCMC	(1)HEW MI, (2)HES SI, (3)GRNN MI, (4) HD MI, (5)WKNN_MI, (6)HOS SI, (7)HES MI, (8)HEW SI, (9)RBF MI, (10) HD SI, (11)KNN MI, (12) RBF SI, (13) EM, (14) WKNN SI, (15) GRNN SI, (16) MLP MI, (17) MLP SI, (18) HOS MI, (19) KNN SI, (20) MS, (21) ZI
3	HOW MI	(1)RBF MI, (2) HD SI, (3)KNN MI, (4) RBF SI, (5) EM, (6) WKNN SI, (7) GRNN SI, (8) MLP MI, (9) MLP SI, (10) HOS MI, (11) KNN SI, (12) MS, (13) ZI
4	GESI	(1)KNN MI, (2) RBF SI, (3) EM, (4) WKNN SI, (5) GRNN SI, (6) MLP MI, (7) MLP SI, (8) HOS MI, (9) KNN SI, (10) MS, (11) ZI
5	HOW SI	(1) GRNN SI, (2) MLP MI, (3) MLP SI, (4) HOS MI, (5) KNN SI, (6) MS, (7) ZI
6	HEW MI, HES SI, GRNN MI, HD MI, WKNN MI, HOS SI	(1) MLP SI, (2) HOM_MI1, (3) KNN SI, (4) MS, (5) ZI
7	HES MI, HEW SI, RBF MI, HD SI, KNN MI, RBF SI, EM, WKNN SI	(1) HOS MI, (2) KNN SI, (3) MS, (4) ZI
8	GRNN SI, MLP MI, MLP SI, HOS MI	(1) KNN SI, (2) MS, (3) ZI
9	KNN SI	(1) MS, (2) ZI
10	MS	(1) ZI
11	ZI	0
With about 20% missing values		
1	GEMI	(1) GESI, (2)HOW SI, (3)HEW MI, (4) GRNN MI, (5) HES MI, (6) RBF MI, (7) HEW SI, (8) MLP MI, (9) WKNN MI, (10) HES SI, (11) HD MI, (12) KNN MI, (13) EM, (14) HOS MI, (15) GRNN SI, (16) RBF SI, (17) WKNN SI, (18) HD SI, (19) KNN SI, (20) MLP SI, (21) HOS SI, (22) MS, (23)ZI
2	MCMC, HOW MI	(1) HES MI, (2)GRNN MI, (3)HEW MI,(4) RBF MI, (5) HEW SI, (6) MLP MI, (7) WKNN MI, (8) HES SI, (9) HD MI, (10) KNN MI, (11) EM, (12) HOS MI, (13) GRNN SI, (14) RBF SI, (15) WKNN SI, (16) HD SI, (17) KNN SI, (18) MLP SI, (19) HOS SI, (20) MS, (21)ZI
3	GESI	(1) HEW SI, (2) MLP MI, (3) WKNN MI, (4) HES SI, (5) HD MI, (6) KNN MI, (7) EM, (8) HOS MI, (9) GRNN SI, (10) RBF SI, (11) WKNN SI, (12) HD SI, (13) KNN SI, (14) MLP SI, (15) HOS SI, (16) MS, (17)ZI
4	HOW SI, HEW MI, GRNN MI	(1) WKNN MI, (2) HES SI, (3) HD MI, (4) KNN MI, (5) EM, (6) HOS MI, (7) GRNN SI, (8) RBF SI, (9) WKNN SI, (10) HD SI, (11) KNN SI, (12) MLP SI, (13) HOS SI, (14) MS, (15)ZI
5	HES MI	(1) HES SI, (2) HD MI, (3) KNN MI, (4) EM, (5) HOS MI, (6) GRNN SI, (7) RBF SI, (8) WKNN SI, (9) HD SI, (10) KNN SI, (11) MLP SI, (12) HOS SI, (13) MS, (14)ZI
6	RBF MI	(1) EM, (2) HOS MI, (3) GRNN SI, (4) RBF SI, (5) WKNN SI, (6) HD SI, (7) KNN SI, (8) MLP SI, (9) HOS SI, (10) MS, (11)ZI
7	HEW SI, MLP MI, WKNN MI	(1) GRNN SI, (2) RBF SI, (3) WKNN SI, (4) HD SI, (5) KNN SI, (6) MLP SI, (7) HOS SI, (8) MS, (9)ZI
8	HES SI, HD MI, KNN MI, EM, HOS MI	(1) RBF SI, (2) WKNN SI, (3) HD SI, (4) KNN SI, (5) MLP SI, (6) HOM_SI1, (7) MS, (8)ZI
9	GRNN SI, RBF SI	(1) WKNN SI, (2) HD SI, (3) KNN SI, (4) MLP SI, (5) HOS SI, (6) MS, (7)ZI
10	WKNN SI	(1) HD SI, (2) KNN SI, (3) MLP SI, (4) HOS SI, (5) MS, (6)ZI
11	HD SI	(1) KNN SI, (2) MLP SI, (3) HOS SI, (4) MS, (5)ZI
12	KNN SI	(1) MLP SI, (2) HOS SI, (3) MS, (4)ZI
13	MLP SI	(1) HOS SI, (2) MS, (3)ZI
14	HOS SI	(1) MS, (2)ZI
15	MS	(1)ZI
16	ZI	0
With about 30% missing values		
1	GEMI	(1)HEW MI, (2) GRNN MI, (3) HOS MI, (4) RBF MI, (5) GESI, (6) MLP MI, (7) HOW SI, (8) HES SI, (9) WKNN SI, (10) MS, (11) KNN MI, (12) KNN SI, (13) EM, (14) HES MI, (15) HD MI, (16) GRNN SI, (17) WKNN MI, (18) HEW SI, (19) HOS SI, (20) HD SI, (21) MLP SI, (22) RBF SI, (23) ZI
2	MCMC, HOW MI	(1) GESI, (2) MLP MI, (3) HOW SI, (4) HES SI, (5) WKNN SI, (6) MS, (7) KNN MI, (8) KNN SI, (9) EM, (10) HES MI, (11) HD MI, (12) GRNN SI, (13) WKNN MI, (14) HEW SI, (15) HOS SI, (16) HD SI, (17) MLP SI, (18) RBF SI, (19) ZI
3	HEW MI	(1) HES SI, (2) WKNN SI, (3) MS, (4) KNN MI, (5) KNN SI, (6) EM, (7) HES MI, (8) HD MI, (9) GRNN SI, (10) WKNN MI, (11) HEW SI, (12) HOS SI, (13) HD SI, (14) MLP SI, (15) RBF SI, (16) ZI
4	GRNN MI	(1) MS, (2) KNN MI, (3) KNN SI, (4) EM, (5) HET_MI1, (6) HD MI, (7) GRNN SI, (8) WKNN MI, (9) HET_SI2, (10) HOM_SI1, (11) HD SI, (12) MLP SI, (13) RBF SI, (14) ZI
5	HOS MI	(1) KNN SI, (2) EM, (3) HES MI, (4) HD MI, (5) GRNN SI, (6) WKNN MI, (7) HEW SI, (8) HOS SI, (9) HD SI, (10) MLP SI, (11) RBF SI, (12) ZI
6	RBF MI	(1) EM, (2) HES MI, (3) HD MI, (4) GRNN SI, (5) WKNN MI, (6) HEW SI, (7) HOS SI, (8) HD SI, (9) MLP SI, (10) RBF SI, (11) ZI
7	GESI	(1) HD MI, (2) GRNN SI, (3) WKNN MI, (4) HEW SI, (5) HOS SI, (6) HD SI, (7) MLP SI, (8) RBF SI, (9) ZI
8	MLP MI	(1) GRNN SI, (2) WKNN MI, (3) HEW SI, (4) HOS SI, (5) HD SI, (6) MLP SI, (7) RBF SI, (8) ZI
9	HOW SI, HES SI, WKNN SI, MS	(1) WKNN MI, (2) HEW SI, (3) HOS SI, (4) HD SI, (5) MLP SI, (6) RBF SI, (7) ZI

Table 6 (*continued*)

Rank	Algorithm	Significantly outperformed algorithms
10	KNN MI, KNN SI	(1) HEW SI, (2) HOS SI, (3) HD SI, (4) MLP SI, (5) RBF SI, (6) ZI
11	EM, HES MI, HD MI, GRNN SI, WKNN MI	(1) HOS SI, (2) HD SI, (3) MLP SI, (4) RBF SI, (5) ZI
12	HEW SI, HOS SI	(1) HD SI, (2) MLP SI, (3) RBF SI, (4) ZI
	HD SI	(1) MLP SI, (2) RBF SI, (3) ZI
13	MLP SI	(1) RBF SI, (2) ZI
14	RBF SI	(1) ZI
15	ZI	0
With about 40% missing values		
1	GEMI	(1) HOW MI, (2) MCMC, (3) HEW MI, (4) MLP MI, (5) GESI, (6) GRNN SI, (7) KNN MI, (8) MLP SI, (9) WKNN SI, (10) KNN SI, (11) HOS SI, (12) ZI, (13) HEW SI, (14) RBF SI, (15) HD MI, (16) EM, (17) HOS MI, (18) HES MI, (19) HES SI, (20) GRNN MI, (21) RBF MI, (22) HD SI, (23) WKNN MI, (24) MS, (25) HOW SI
2	HOW MI	(1) HOS SI, (2) ZI, (3) HEW SI, (4) RBF SI, (5) HD MI, (6) EM, (7) HOS MI, (8) HES MI, (9) HES SI, (10) GRNN MI, (11) RBF MI, (12) HD SI, (13) WKNN MI, (14) MS, (15) HOW MI
3	MCMC	(1) ZI, (2) HEW SI, (3) RBF SI, (4) HD MI, (5) EM, (6) HOS MI, (7) HES MI, (8) HES SI, (9) GRNN MI, (10) RBF MI, (11) HD SI, (12) WKNN MI, (13) MS, (14) HOW SI
4	HEW MI	(1) HEW SI, (2) RBF SI, (3) HD MI, (4) EM, (5) HOS MI, (6) HES MI, (7) HES SI, (8) GRNN MI, (9) RBF MI, (10) HD SI, (11) WKNN MI, (12) MS, (13) HOW SI
5	MLP MI	(1) RBF MI, (2) HD SI, (3) WKNN MI, (4) MS, (5) HOW SI
6	GESI, GRNN SI, KNN MI, MLP SI, WKNN SI, KNN SI, HOS SI, ZI	(1) HD SI, (2) WKNN MI, (3) MS, (4) HOW SI
7	HEW SI, RBF SI	(1) WKNN MI, (2) MS, (3) HOW SI
8	HD MI, EM, HOS MI	(1) MS, (2) HOW SI
9	HES MI, HES SI, GRNN MI, RBF MI, HD SI, WKNN MI, MS	(1) HOW SI
10	HOW SI	0
With about 50% missing values		
1	GEMI	(1) HES MI, (2) GESI, (3) HEW SI, (4) RBF MI, (5) GRNN SI, (6) MLP SI, (7) KNN SI, (8) MCMC, (9) RBF SI, (10) HEW MI, (11) HES SI, (12) HD MI, (13) KNN MI, (14) EM, (15) HOW SI, (16) GRNN MI, (17) HD SI, (18) MLP MI, (19) WKNN MI, (20) HOS SI, (21) MS, (22) HOS MI, (23) ZI
2	HOW MI	(1) KNN MI, (2) EM, (3) HOW SI, (4) GRNN MI, (5) HD SI, (6) MLP MI, (7) WKNN MI, (8) HOS SI, (9) MS, (10) HOS MI, (11) ZI
3	HES MI	(1) GRNN MI, (2) HD SI, (3) MLP MI, (4) WKNN MI, (5) HOS SI, (6) MS, (7) HOS MI, (8) ZI
4	GESI, HEW SI, RBF MI, GRNN SI, MLP SI, KNN SI	(1) MS, (2) HOS MI, (3) ZI
5	MCMC, RBF SI, HEW MI, HES SI, HD MI, KNN MI, EM	(1) HOS MI, (2) ZI
6	HOW SI, GRNN MI, HD SI, MLP MI, WKNN MI, WKNN SI, HOS SI, MS	(1) ZI
7	ZI	0
With about 60% missing values		
1	GEMI	(1) ZI, (2) GESI, (3) HOW MI, (4) HOW SI, (5) GRNN MI, (6) GRNN SI, (7) MLP MI, (8) WKNN MI, (9) RBF SI, (10) HOS SI, (11) MS, (12) HOS MI
2	MCMC, HEW MI, HES MI, HEW SI, HES SI, HD MI, RBF MI, HD SI, KNN MI, EM, MLP SI, WKNN SI, KNN SI, ZI	(1) HOS MI
3	GESI, HOW MI, HOW SI, GRNN MI, GRNN SI, MLP MI, WKNN MI, RBF SI, HOS SI, MS, HOS MI	0
With about 70% missing values		
1	GEMI	(1) HOW MI, (2) MCMC, (3) HEW MI, (4) MLP MI, (5) GESI, (6) GRNN SI, (7) KNN MI, (8) MLP SI, (9) WKNN SI, (10) KNN SI, (11) HOS SI, (12) ZI, (13) HEW SI, (14) RBF SI, (15) HD MI, (16) EM, (17) HOS MI, (18) HES MI, (19) HES SI, (20) GRNN MI, (21) RBF MI, (22) HD SI, (23) WKNN MI, (24) MS, (25) HOW SI
2	HOW MI, MCMC, HEW MI, MLP MI, GESI, GRNN SI, KNN MI, MLP SI, WKNN SI, KNN SI, HOM_SI1, ZI, HEW SI, RBF SI, HD MI, EM, HOS MI, HES MI, HES SI, GRNN MI, RBF MI, HD SI, WKNN MI, MS, HOW SI	0

value estimates is not a good measure for reflecting the goodness of imputations since no top imputation algorithm in terms of missing value estimation accuracy achieved good classification results. It is also interesting to note that the single imputation algorithms achieved the best missing value estimation accuracy (i.e. the point estimation accuracy of missing values), while the multiple imputation algorithms achieved the best classification accuracy of the response variable. This is because multiple imputation algorithms simulate the entire joint distribution of the unknown values. In contrast, single imputation algorithms estimate only the conditional mean of the missing value so that the width of the confidence interval is zero at the imputed value that leads to over-fitting or over-optimization. For all these reasons, single imputation algorithms typically offer higher accuracy in the determination of the missing data, than do any of the multiple imputation algorithm; but provides lower accuracy in downstream analyses.

- The CPU cost of all algorithms increases linearly in the range of a 5–50% missing rate (Table H6). Surprisingly, beyond this range, the computational time drops dramatically. This is perhaps because there is so little information available about missing values that the algorithms quickly converge to suboptimal solutions.

In our study, the computational costs were very large both for the proposed algorithms and for the conventional algorithms except Mean Substitution and Zero Imputation methods. This is because, in our study, for fair comparison and evaluation, all imputation algorithms employ SAGA for feature subset selection.

All algorithms fit imputation models using the iterative EM-style training approach.

SI algorithms fit one model (for the conditional mean) for each missing value while MI algorithms fit 30 pairs of models (one for the conditional mean and one for the conditional variance) for each missing value. Hence, we expect the MI algorithms to be approximately 30 times more computationally intensive than SI algorithms. However, in reality, the computational cost does not increase so much to the MI algorithms because SAGA is used for selecting feature subsets only once during the model formulation—in the construction of prototype models, not in the construction of replica models.

6. Summary and conclusions

We have presented a multiple imputation algorithm GEMI and a single imputation algorithm GESI. Both algorithms use the generalized regression neural network ensemble. We tested new algorithms on 98 synthetic and real-world datasets. All simulation results show the advantages of GEMI as compared with the conventional algorithms. However, we note that GEMI has heavy memory storage requirements and is expensive computationally. GEMI draws multiple samples from the training set in order to calculate the conditional posterior distribution of the missing value and then the initial training set is augmented several times. GEMI is an instance-based algorithm that stores all instances of the training sample in a memory in order to use them when needed. In addition, GEMI employs a relatively expensive feature subset selection algorithm SAGA to identify not only the good subsets of features, but also an optimal subset of ensemble members. Moreover, GEMI resorts to an EM-style iterative procedure to refine the imputation models. Thus, fitting a joint distribution and generating multiple imputations using GEMI can greatly increase the computational requirements, both in terms of processor speed and storage.

To ensure feasibility with respect to time and resource constraints, one can play with parameter values (the parameters of SAGA and the parameter m) accepting the trade-off between precision and cost. It is found that GEMI is better than other multiple imputation (and single imputation) algorithms, whereas GESI is better than other single imputation algorithms. Therefore, for a given value of m, GEMI will perform better than the existing ones no matter what value the parameter m takes (however, to reduce the risk of overfitting, the value of this parameter must be set at least to 2). Similarly, no matter how much time we give SAGA, SAGA can help us choose a better ensemble, since in our feature subset selection experiments [31], SAGA came up with better feature subset solutions compared to conventional search algorithms within all given time frames.

Proper handling of missing values is essential in all analyses. Existing fast but inaccurate imputation methods can hinder downstream analysis of the dataset as they frequently destroy the original distribution of the dataset. Although using GEMI is relatively computationally expensive with associated intensive memory requirements, the significantly better results justify its use. The generation of high-quality imputations always has a greater priority than computational complexity. Besides, modern computers are powerful enough to handle this type of computationally intensive application, for all but the largest datasets.

Appendix A. Generalized regression neural networks (GRNN)

GRNN is a simple, yet very powerful learning algorithm. In GRNN [30] each observation in the training set forms its own cluster [142]. GRNN is an instance-based algorithm. In GRNN each observation in the training set forms its own cluster. When a new input pattern $x=(x_1,\ldots,x_n)$ is presented to the GRNN for the prediction of the output value, each training pattern (prototype pattern) $y_i=(y_{i1},\ldots,y_{in})$ assigns a membership value h_i to x based on the Euclidean distance d, where

$$d=d(x,y_i)=\sqrt{\sum_{j=1}^{m}(x_j-y_{ij})^2} \tag{6}$$

and

$$h_i=\exp\left(-\frac{d^2}{2\sigma^2}\right) \tag{7}$$

n is the total number of features in the study. x_j is the value of the jth feature of the presented pattern (features can be multi-valued or not). y_{ij} is the value of the jth feature of the ith prototype pattern and σ is the smoothing function parameter. We found that the performance of GRNN is not very sensitive to the exact setting of the parameter (σ). We arbitrarily set each centre's width to two times of the average distance to 20 nearest neighbours.

Finally, GRNN calculates the output value z of the pattern x as in Eq. (8). The predicted output of the GRNN for the pattern x is the weighted average of the outputs of all prototype patterns. GRNN can handle continuous output variables and categorical output variables with two categories: event of interest (coded as '1') or not (coded as '0'):

$$z=\frac{\sum_i(h_i\times \text{output of } y_i)}{\sum_i h_i} \tag{8}$$

If the output variable is binary, then GRNN calculates the probability of event of interest. If the output variable is continuous, then it estimates the value of the variable.

Appendix B. Overview of SAGA

The proposed algorithm GEFTS uses an improved algorithm SAGA [31] for selecting an optimal subset of features, both for base GRNNs and the combiner GRNN. SAGA uses GRNN for assessing the fitness of feature subsets. SAGA works in three stages. During stage 1, SAGA applies the simulated annealing (SA) algorithm on 100 randomly selected possible solutions. SA leads to global exploration of search space without getting trapped into a local minimum. If the best solution does not improve 100 consecutive generations, the first stage is terminated. During stage 2, SAGA applies the genetic algorithm (GA) on the 100 best-to-date solutions found by the SA. A total of 50 pairs are picked from the chromosome pool using linear ranking selection. Selection is done "with replacement" meaning that the same chromosome can be selected more than once to become a parent. Each pair creates two offspring using the half uniform crossover scheme (HUX) and then the parents die. In HUX, exactly half of the non-matching parent's genes are swapped. Due to selection of the fittest chromosomes, the crossover and a very low mutation rate (0.0001), GA converges quickly to a near optimal solution. The second stage ends if the best solution does not improve in 100 consecutive generations. In the final stage, SAGA refines the search by hill-climbing on the 10-best-to-date solutions. The pseudocodes of SA, GA and HC are given below.

B.1. Pseudocode of the simulated annealing (SA)

// Initialization section

Step 1: Encode possible solutions in binary strings, where 1 indicates the presence of the feature (or the base classifier) and 0 indicates its absence.

Step 2: Set the initial temperature (T_i): $T_i=100$
Step 3: Set the current temperature (T_c): $T_c=T_i$
Step 4: Initialize population: Randomly select 100 feature subset $I(=I(1{:}100))$ solutions from the solution space for initial population.
Step 5: Estimate the prediction accuracy of each solution.
Step 6: Evaluate the fitness (fitness score is the prediction accuracy as a fraction, not a percentage) of each solution using 10-fold cross validation**:** Based on binary string of each solution I, extract a new dataset D_{new} from the (normalized) original dataset D. Evaluate the fitness scores $E_o(=E_o(1{:}100))$ of feature subsets using GRNN and store the information (feature subset solutions with feature score) in database.// Iterative Section
Step 7: For all current feature subset vectors $I(=I(1{:}100))$ change the bits of vectors with probability

$$p_m(=p_m(1:100)) : p_{mi}=1-E_{oi}$$

where p_{mi} represents the mutation rate of bits within the ith solution and E_{oi} represents the fitness score of the ith solution.
Step 8: Evaluate the fitness $E_n(=E_n(1{:}100))$ of the new candidate solutions if not already evaluated (check the database for fitness scores).
Step 9: Determine if this new solution is kept or rejected and update the database:

- If $E_n \geq E_o$, the new solution is accepted. The new solution replaces the old solution and E_o is set to E_n.
- If $E_n < E_o$, calculate the Boltzmann acceptance probability P_{accept}: $P_{accept}=\exp(-(E_o-E_n)/T_c)$.
- Generate a random number between 0 and 1. If P_{accept} is greater than or equal to the random number, the new solution is accepted and it replaces the old one: $E_o=E_n$.

Step 10: Update the effective temperature T_c:

- If the fitness of the best solution does not improve: $T_c=T_c-1$.
- If the fitness of the best solution improved: $T_c=T_i$

Step 11: If the effective temperature is greater than or equal to zero, return to Step 7. Otherwise the run is finished.

B.2. The pseudocode of GA

Step 1: Construct a chromosome pool of size 100 with the 100 fittest chromosomes from the list of feature subset solutions evaluated so far by the SA.
Step 2: Select 50 pairs of chromosomes with replacement using rank-based selection strategy.
Step 3: Perform crossover between the chromosomes using the half uniform Crossover scheme (HUX). In HUX, half of the non-matching parents' genes are swapped.
Step 4: Kill the parent solutions.
Step 5: Mutate offspring with probability 0.0001.
Step 6: Evaluate the fitness of the offspring provided if it has not already been evaluated. Update the database and estimate the time left.
Step 7: Go back to Step 2 if the best solution does not improve in the last 100 runs.

B.3. Pseudo-code of Hill-climbing algorithm (HC)

Step 1: Select the best-to-date solution.
Step 2: Create N new candidate solutions from the selected solution by changing only one bit (feature) at a time. N denotes the total number of features in the feature space.
Step 3: Evaluate the new solutions if they are not evaluated before and update the database. Replace the previous solution by the new solution(s) if they are better than the previous solution.
Step 4: Go back to step 2 and perform the hill climbing on each of the accepted new solutions. Repeatedly apply the process from steps 2 to 3 on selected solutions as long as the process is successful in finding improved solutions in every repetition and as long as the time is available.
Step 5: Update the database and update the time available.
Step 6: Select the next best-to-date solution from the database and go back to step 2. Thus perform hill-climbing on the 10 best-to-date solutions.

Appendix C. The impact of different parameter settings

See Tables C1 and C2.

Appendix D. Description of statistical tests

D.1. The Friedman two-way analysis of variance by ranks

Null hypothesis: The performance of k different algorithms have the same rank totals.

Alternative hypothesis: The performances of k different algorithms have significantly different rank totals.

The Friedman test determines whether the rank totals for each algorithm differ significantly from the values which would be expected by chance. To do this test, we compute the value of the statistic which we shall denote as F_r

$$F_r=\left[\frac{12}{Nk(k+1)}\sum_{j=1}^{k}R_j^2\right]-3N(k+1) \tag{9}$$

Table C1
The performance of different smoothing factor parameter values (standard deviations in parentheses).

Rank (based on statistical significance)	Each centre's width	Classification accuracy (%)
4	(Euclidean distance to the nearest member)*0.5	79 (12)
3	Euclidean distance to the nearest member	85 (8)
3	(Euclidean distance to the nearest member)*1.5	87 (7)
3	(Euclidean distance to the nearest member)*2	88 (7)
3	(Euclidean distance to the nearest member)*2.5	84 (8)
4	(Euclidean distance to the nearest member)*3	81 (12)
1	(Average Euclidean distance to 5 nearest members)*2	95 (4)
1	(Average Euclidean distance to 10 nearest members)*2	96 (4)
1	(Average Euclidean distance to 20 nearest members)*2	97 (3)
1	(Average Euclidean distance to 30 nearest members)*2	96 (5)
1	(Average Euclidean distance to 40 nearest members)*2	95 (5)
2	(Average Euclidean distance to 50 nearest members)*2	92 (7)
3	(Average Euclidean distance to 60 nearest members)*2	88 (10)
4	(Average Euclidean distance to 70 nearest members)*2	84 (11)
5	(Average Euclidean distance to 80 nearest members)*2	75 (12)
5	(Average Euclidean distance to 90 nearest members)*2	72 (14)
5	(Average Euclidean distance to 100 nearest members)*2	72 (17)

Table C2
Impact of number of imputations (m) on the performance of GEMI (standard deviations in parentheses).

m	Classification accuracy (%)	Rankings (based on test results)	m	Classification Accuracy (%)	Rankings (based on test results)	m	Classification accuracy (%)	Rankings (based on test results)
With about 5% missing values			**With about 10% missing values**			**With about 20% missing values**		
1	84 (7)	4	1	71 (13)	6	1	70 (18)	6
2	96 (4)	3	2	89 (6)	5	2	84 (9)	5
5	98 (3)	2	5	92 (3)	4	5	87 (8)	4
10	96 (4)	1	10	96 (3)	3	10	88 (8)	3
20	96 (2)	1	20	96 (3)	2	20	90 (5)	2
30	98 (2)	1	30	96 (3)	1	30	94 (5)	1
50	96 (3)	1	50	96 (3)	1	50	94 (3)	1
100	99 (3)	1	100	97 (3)	1	100	95 (3)	1
200	98 (3)	1	200	97 (3)	1	200	95 (3)	1
With about 30% missing values			**With about 40% missing values**			**With about 50% missing values**		
1	61 (19)	6	1	54 (21)	4	1	53 (20)	6
2	78 (11)	4	2	82 (13)	4	2	68 (14)	5
5	82 (12)	5	5	82 (12)	3	5	71 (15)	3
10	82 (13)	4	10	85 (8)	2	10	76 (9)	4
20	85 (9)	3	20	87 (8)	3	20	75 (7)	3
30	91 (7)	2	30	91 (6)	2	30	82 (7)	2
50	90 (6)	2	50	92 (5)	2	50	82 (7)	1
100	92 (6)	1	100	93 (5)	1	100	83 (6)	1
200	93 (6)	1	200	93 (5)	1	200	83 (6)	1
With about 60% missing values			**With about 70% missing values**			**With about 75% missing values**		
1	54 (19)	4	1	53 (20)	2	1	50 (15)	1
2	60 (16)	3	2	55 (20)	2	2	46 (12)	1
5	67 (12)	2	5	54 (18)	1	5	46 (18)	1
10	62 (16)	2	10	63 (12)	1	10	50 (16)	1
20	70 (9)	1	20	66 (12)	1	20	51(14)	1
30	76 (7)	1	30	68 (11)	1	30	48 (14)	1
50	80 (8)	1	50	68 (11)	1	50	52 (14)	1
100	80 (7)	1	100	73 (11)	1	100	52 (14)	1
200	80 (7)	1	200	73 (11)	1	200	52 (15)	1

where N is the number of datasets, k is the number of search algorithms, R_j is the sum of ranks of the jth algorithm, $\sum_{j=1}^{k} R_j^2$ = sum of the squares of the sums of ranks over all algorithms.

Appendix table M in [33] gives the probabilities associated with values of F_r as large or larger than the tabled values for various values of N and k. If the observed value of F_r is larger than the tabled value of F_r at the chosen significance level, then null hypothesis may be rejected in favour of alternative hypothesis.

When the obtained value of F_r is significant, it indicates that at least one of the algorithms differ from at least one other algorithm. It does not tell the researcher which one is different, nor does it tell the researcher how many of the algorithms are different from each other. For these answers, we performed the statistical test "comparisons of groups or conditions with a control" for each pair of algorithms. We discuss this test in the following section.

D.2. Comparisons of groups or conditions with a control

Null hypothesis: The performances of two algorithms (algorithms 1 and 2) are the same.

Alternative hypothesis: The performances of two algorithms (algorithms 1 and 2) are not the same.

We can test the significance of differences between two algorithms by using the following inequality. That is, if

$$|R_1 - R_2| \geq q(\alpha, \pm c)\sqrt{\frac{Nk(k+1)}{6}} \tag{10}$$

where R_1 is the rank total of the algorithm 1; R_2 is the rank total of the algorithm 2; N is the total number of datasets, $k$is the total number of search algorithms ranked, $c=k-1$; $\alpha=0.05$. α represents the level of significance in statistical tests. Values of $q(\alpha, \pm c)$ are given in Appendix Table A_{III} in [33].

If the value of $|R_1 - R_2|$ exceeds the value of $q(\alpha, \pm c)\sqrt{(Nk(k+1))/6}$, there is a statistically significant difference between the two algorithms.

Appendix E. Implementation of conventional missing data imputation algorithms

We compared the proposed missing data imputation algorithm with a number of single imputation algorithms as well as several multiple imputation algorithms.

We treat the variable with missing value as target, the remaining variables as predictors. The candidate input variables include both the main variables and corresponding indicator variables. We explain how we implemented these algorithms in this section.

E1. *Single imputation* (*SI*) *algorithms*.

E2. *Zero imputation* (*ZI*) replaces the missing values with zero.

E3. *Mean substitution* (*MS*) fills in the missing values by their variable means.

E4. *Hot deck imputation* (HD) works in the following two stages:

Step 1: Find the closest donors for the missing value from complete records using the Euclidean distance matching function.

Step 2: Substitute the most similar case's value for the missing value.

E5. *K-nearest neighbours algorithm* (*KNN*) imputes missing values by the average value of the K nearest patterns using the following equation:

$$x_{ij} = \frac{\sum_{k=1}^{K} x_{kj}}{k} \tag{11}$$

where x_{ij} represents a missing value in the jth variable of the ith instance. K is the number of nearest neighbours and x_{kj} is the value of the jth variable of the kth nearest neighbour.

E6. *Weighted K-nearest neighbours* (*WKNN*) *algorithm* replaces missing values with a weighted average of the K-nearest neighbours, as presented in the following equation. Let us assume that the value of the jth variable of the ith instance (x_{ij}) is missing.

Using WKNN algorithm, x_{ij} is replaced by

$$x_{ij} = \frac{\sum_{k=1}^{K} w_k x_{kj}}{\sum_{k=1}^{K} w_k} \tag{12}$$

where $w_k = 1/d_{ik}$

Here, w_k is the weight associated to the kth nearest neighbour. x_{kj} is the value of the jth variable of the kth nearest neighbour. d_{ik} is the Euclidean distance between the ith pattern (the instance with the missing value) and the kth nearest neighbour. K is the number of nearest neighbours. In other words, the weight w_k of the kth nearest donor is equal to the reciprocal of its Euclidean distance to the instance with missing values.

E7. *Expectation maximization* (*EM*) is a procedure for parameter estimation in the presence of missing data. It is an iterative two-step (E-step and M-step) process.

The *E* (*Expectation*) *step* fills in the missing values using estimated values for the model parameters (initially random values are assigned to these parameters).

Then, the *M* (*Maximization*) *step* re-estimates the parameters by using the observed and imputed values to maximize the log-likelihood of the model.

The algorithm iterates from E to M steps until the log-likelihood converges to a stationary point. The implementation of EM assumes all attributes to be independent and normally distributed.

E8. *Single imputation with neural network-based algorithms*: Single imputation algorithms create a model for predicting the conditional mean of the missing value. The candidate input variables include the main variables and the indicator variables. The output variable is the variable with missing values. Three neural network models (MLP, RBFN, and GRNN) were designed for the imputation of missing data. We also tested four neural network ensemble models: heterogeneous ensemble with simple averaging for single imputation (HES SI), heterogeneous ensemble with weighted averaging for single imputation (HEW SI), homogeneous ensemble with simple averaging for single imputation (HOS SI), and homogeneous ensemble with weighted averaging for single imputation (HOW SI). The members in heterogeneous ensembles (HES SI and HEW SI) are a MLP, a RBFN and a GRNN, whereas the members in homogeneous ensembles (HOS SI and HOW SI) are GRNNs. Like EM, neural networks (MLP, RBFN, and GRNN) and neural network ensembles (HES SI, HEW SI, HOS SI, and HOW SI) impute missing values by two iterative steps: (1) imputing missing values based on initial estimates of model parameter values and (2) updating model parameters. Currently, many powerful and flexible neural network modelling software packages are available. Hence in this section, we only discuss how the free parameters of single and ensemble neural network models are fixed.

For each neural network topology, 100 networks with different initial weight configurations were trained and the network with the best performance on the validation set was chosen.

E8.1. *Tuning of the free parameters of MLP*

- *Number of input nodes* (*input variables*): The number of nodes in the input layer is decided by SAGA.
- *Number of hidden layers*: Empirical studies suggest that generally for most applications one hidden layer is sufficient. The new findings also suggest that the risk of overfitting increases with the number of hidden layers. After considering the pros and cons, we chose one hidden layer for our MLP models.
- *Number of hidden nodes*: The number of nodes in the hidden layer is equal to the number of hidden nodes plus one bias node. Too few or too many hidden nodes can cause the MLP to underfit or overfit during training. In most situations, there is no way to determine the best number of hidden nodes except through trial, error and observation. We determine the number of hidden nodes with the trial and error procedure.
- *Types of transfer functions:* A transfer function ensures that the values in a network remain within a reasonable range. The transfer functions in different layers may or may not be identical, but the transfer functions for all nodes in the same layer should be identical, so that all input values for a hidden or output node are within the same range. We scaled data in the range of 0–1. Hence, the forecasted values are assumed to be bounded in the range of 0–1. To satisfy this requirement, the logistic-sigmoid function was chosen as the activation function for the output node. The main purpose of the hidden node transfer functions is to introduce nonlinearity into the network. Common hidden node transfer functions are logistic function, tangent hyperbolic function (tanh), and Gaussian function. We tried each of these three functions individually during our tests and in all cases the performance of the network was pretty much the same. In this study, tanh is selected as the activation function of hidden nodes.

E8.2. *Parameter specification for radial basis function neural networks* (*RBFN*): Optimum number of hidden nodes: are determined using a trial-and-error approach. Each hidden node corresponds to a prototype pattern. These prototype patterns are selected using the K-medoid clustering algorithm (described below). The widths of the radial basis functions are optimized using the real-valued particle swarm optimization (PSO) algorithm, since the PSO was originally developed for real-valued spaces (described in section E10).

We adjust connection weights of the RBFN using the back-propagation technique.

E.1. K-Medoid clustering algorithm

Step 1: Begin with a decision on the value of K=number of clusters.

Step 2: Take the first K training input patterns as cluster centres (centroids). Assign each of the remaining (N-K) training input patterns to cluster with the nearest centroid. After each assignment, re-compute the centroid of the gaining cluster. The centroid is the input pattern with the minimum average Euclidean distance to all members in the cluster.

Step 3: Take each sample in sequence and compute its (Euclidean) distance from the centroid of each of the clusters. If a sample is not currently in the cluster with the closest centroid, switch this input pattern to that cluster and update the centroid of the cluster gaining the new input pattern and the cluster losing the input vector.

Step 4: Repeat step 3 until convergence is achieved, that is until a pass through all training patterns causes no new assignments.

E9. *Ensemble neural networks*: ANN ensemble consists of several individually trained ANN classifiers (base classifiers) that are jointly used to solve a problem. In our experiments, two types of ensembles—homogeneous and heterogeneous ensembles are

constructed. A heterogeneous ensemble is a collection of different neural networks (ERNN, GRNN, MLP, and RBFN) that together "vote" on a given example. All the individual networks in the heterogeneous ensemble are trained on the same training data, with the same predictors. In contrast, a homogeneous ensemble is a collection of the same kind of neural networks trained using different feature subsets as opposed to only one feature subset used in a heterogeneous ensemble model. For generating homogeneous ensemble models, we used multiple GRNNs each trained using different feature subsets. Each ensemble member tries to predict the response variable. The base classifiers of heterogeneous ensemble models constructed for the substitution of missing values include: GRNN, MLP, and RBFN.

The outputs of ensemble members are fused together to get the final decision. Majority and weighted majority voting are common methods for combining the outputs of ensemble members. In our study, we evaluated four types (two homogeneous and two heterogeneous) of neural network ensembles: (1) heterogeneous ensemble with majority voting, (2) heterogeneous ensemble with weighted majority voting, (3) homogeneous ensemble with majority voting, and (4) homogeneous ensemble with weighted majority voting.

In majority voting, the final prediction of the ensemble on each test data point is an average of the predictions of all ensemble members as follows:

$$Y_i = \frac{\sum_{j=1}^{n} \widehat{y}_{ij}}{n} \tag{13}$$

where Y_i is the actual output of the ith pattern, $\widehat{y}_{ij}$ is the output of the ith pattern predicted by the jth member, and n is the total number of base classifiers.

In weighted majority voting, each ensemble member votes with its confidence. The final output of the ensemble was calculated using the following equation:

$$Y_i = \frac{\sum_{j=1}^{n} (w_j \widehat{y}_{ij})}{\sum_{j=1}^{n} w_j} \tag{14}$$

where w_j is the weight with which the jth ensemble member participates in the final output and $\widehat{y}_{ij}$ is the output of the ith pattern predicted by the jth member.

The vote weights of the base classifiers were optimized by a real-valued PSO. The vote weights are assigned to base classifiers based on the global best-fitted combination. The optimization process is described in detail here.

We normalize each variable to the range [0,1].

E10. *Pseudo-code of real-valued PSO for optimizing the parameter set*

Step 1: Specify parameters

- *Range of weights*: Constrain ranges of values of parameters of interest to be within 0 and 1.
- *Population size*: The swarms were set to 10 particles.
- *Position of each particle*: The ith particle of the swarm at time t can be represented by a position vector. $X_i(t) = [X_{i1}(t), X_{i2}, X_{i3}, X_{i4}, \ldots]$, where each bit represents the position (i.e. value) of a particular parameter of a paricle (parameter vector) and each bit is a real value in the interval [0,1]. We randomly initialize the position of each particle.
- *Maximum and minimum velocities*: We set maximum velocity ($V_{\max}$) of any particle to be 1/3 or 0.33 and minimum velocity ($V_{\min}$) to be 0.05. Velocity determines the change in a parameter value at a particular time t.
- *Velocity of each particle*: The velocity of the ith particle at time t is denoted by $V_i(t) = [V_{i1}(t), V_{i2}(t), V_{i3}(t), V_{i4}(t)]$ where each bit represents the velocity of a particular parameter of a particle and each bit is a real value in the interval [0.05–0.33]. Particle velocities are initially randomly determined.

Step 2: Evaluate the fitness F (prediction accuracy) of each particle by training a machine learning algorithm using the particle's current position $X_i(t)$. The prediction accuracy was estimated by 10-fold cross validation.Let $F(X_i)$ is the fitness score of the ith particle.$F(X_i)$ is the prediction accuracy of ith particle as a fraction, where $0 \le F(X_i) \le 1$.

Step 3: The swarm was initialized randomly with each particle's personal best position ($P_{i,best}$) being the same as its current position. Compare the fitness of each particle $F(X_i)$ to its best fitness so far $F(P_{i,best})$; and update each particle's best position. If $F(X_i) > F(P_{i,best})$, then $F(P_{i,best}) = F(X_i)$, and $P_{i,best} = X_i$

Step 4: Update the global best particle (P_{gbest}) and its fitness ($F(P_{gbest})$):If $F(X_i) > F(P_{gbest})$, then $F(P_{gbest}) = F(X_i)$ and $P_{gbest} = X_i$.

Step 5: Change the velocity of each bit of the particle according to

$$V_{ij}(t+1) = wV_{ij}(t) + c_1 r_1 (P_{ij.best} - X_{ij}) + c_2 r_2 (P_{gbest} - X_{ij}) \tag{15}$$

where $w = 1 - (T_{spent}/T_{max})$

If $V_{ij}(t+1) > V_{max}$, then $V_i(t+1) = V_{max}$

If $V_{ij}(t+1) < V_{min}$, then $V_i(t+1) = V_{min}$

where $V_{ij}(t+1)$ and $V_{ij}(t)$ denote velocities of jth bit for the ith particle at time $(t+1)$ and t, respectively. w is the inertia weight which shows the effect of previous velocity vector on the new vector. $T_{\max}$ denotes the total number of iterations and T_{spent} denotes the number of iterations performed so far. r_1 and r_2 are two random numbers between (0, 1). c_1 and c_2 are two positive constants: $c_1 = c_2 = 2$.

Step 6: Move each particle to the new position using

$$X_{ij}(t+1) = X_{ij}(t) + V_{ij}(t+1) \tag{16}$$

If $X_{ij}(t+1) > 1$; then $X_{ij} = 1$ [since the maximum weight cannot be greater than 1]If $X_{ij}(t+1) < 0$, then $X_{ij}(t+1) = 0$ [since the minimum parameter value cannot be less than 0].

Step 7: Go to step 2, and repeat until stopping criterion is met.

E11. *Multipleimputation*

We implemented the following multiple imputation algorithms: Markov chain Monte Carlo (MCMC—a multiple-imputation version of the EM algorithm), multilayer perceptron with multiple imputation (MLP MI), radial basis function networks with multiple imputation (RBFN MI), generalized regression networks with multiple imputation (GRNN MI), HES MI (a multiple-imputation version of the HES SI), HEW MI (a multiple-imputation version of the HEW SI), HOS MI (a multiple-imputation version of the HOS SI), HOW MI (a multiple-imputation version of the HOW SI), hot deck multiple imputation (HD MI), K-nearest neighbours algorithm with multiple imputation (KNN MI) and weighted K-nearest neighbours algorithm with multiple imputation (WKNN MI). A commonly practiced multiple imputation (MI) analysis consists of three steps: imputation, analysis and pooling.

Imputation: In this step, each missing value is imputed for several (M) times, which yields m complete datasets. In MCMC, MLP MI, and RBFN MI, multiple imputations are generated by randomly selecting M sets of initial parameter estimates. Starting at different initial parameter guesses will generally lead to different local optimal solutions. In GRNN MI, HOS MI, HOW MI, HES MI, HEW MI, KNN MI and WKNN MI, M new training sets are randomly extracted from the original training set, each comprising 70% training examples. Different training sets will lead to different imputation models. M different models will lead to different complete dataset. In HD MI, M donors are randomly chosen for each missing value from the pool of potential donors that will lead to M different complete datasets. Like EM, each of

the following multiple imputation algorithms :MCMC, MLP MI, RBFN MI, GRNN MI, HES MI, HEW MI, HOS MI, and HOW MI is an iterative process that alternatively fills in missing values and makes inferences about the unknown parameters. However, these algorithms do this in a stochastic or random fashion.

The *Imputation I-step*: Given an estimated mean vector and covariance matrix of parameters, the *I*-step simulates the missing values for each data point independently by randomly drawing parameters from their conditional distribution.

The *posterior P-step*: Given a complete dataset obtained in the previous step, the mean vector and covariance matrix are recomputed. The new mean vector and covariance matrix are then used in the next *I*-step.

The above two steps are iterated until the mean vector and covariance matrix stabilize.

Analysis: Each of the M completed datasets are analyzed. This step results in the m analysis results. For example, with m imputations, m different sets of mean and variance for a missing value can be computed. Let us suppose that $\widehat{Y}_i$ and $\hat{U}_i$ are the mean and variance estimates from the ith imputed dataset, $i=1,2,\ldots,m$.

Multiple imputation algorithms construct two models—*Model* 1 for predicting the conditional mean ($\widehat{Y}_i$) of the missing value and *Model* 2 for predicting the conditional variance ($\hat{U}_i$) of the missing value. The output variable in *Model* 1 is the variable with missing values, whereas the output variable in *Model* 2 is the squared residuals of the fitted *Model* 1. The predictor variables of both models are the main variables and the indicator variables.

Pooling: The m analysis results are integrated into a final result (i.e. the vector of means and the variance–covariance matrix of model parameters). Simple rules exist for combining the m analysis results

$$\overline{Y}=\frac{1}{M}\sum_{i=1}^{M}\widehat{Y}_i \tag{17}$$

Let us suppose that $\overline{U}$ is the within-imputation variance, which is the average of the M complete data-estimates:

$$\overline{U}=\frac{1}{M}\sum_{i=1}^{M}\hat{U}_i \tag{18}$$

And B is the between-imputation variance

$$B=\frac{1}{(M-1)}\sum_{i=1}^{M}(\widehat{Y}_i-\overline{Y})^2 \tag{19}$$

Then the variance estimate associated with $\overline{Q}$ is the total variance [11]

$$T=\overline{U}+\left(1+\frac{1}{M}\right)B \tag{20}$$

We replicate each record of the original dataset $m$times. We then impute the missing values setting them to $\overline{Y}+\sqrt{T}R$ where R is a pseudo random number drawn from a normal distribution with mean 0 and standard deviation 1.

Appendix F. Description of datasets

New real world dataset (*smoking dataset*): We received a three stage cross sectional survey data on the smoking habits of teenagers from the centre for tobacco control research at the University of Stirling and Open University. The data were collected from Scotland, England, Northern Ireland and Wales in three survey stages: stage1 in 1999, stage 2 in 2002 and stage 3 in 2004. The response variable is a binary variable (1=smoker, 0=non-smoker). Explanatory variables include socio-demographic characteristics of respondents, their knowledge and attitudes towards tobacco promotion of all sorts and their smoking knowledge, attitudes and behaviour. This smoking dataset contains 285 features, 3321 instances but has a large number of missing values. This dataset contains about 37% missing values. Among the respondents, an overall proportion of 11% (355 respondents) are smokers. We applied our proposed missing data imputation algorithm (GEMI) to replace missing values (details are available in Section 5). We did not add artificial features to this dataset.

Public real-world datasets: The public real-world datasets are obtained from UCI Machine Learning Repository [145]. The UCI datasets on which we tested the algorithms are: (1) Abalone, (2) Acute Inflammations, (3) Adult, (4) Annealing, (5) Arcene, (6) Arrhythmia, (7) Automobile, (8) Auto MPG, (9) Balance Scale, (10) Balloons, (11) Blood Transfusion Service Center, (12) Breast Cancer Wisconsin (Diagnostic), (13) Breast Cancer Wisconsin (Prognostic), (14) Car Evaluation, (15) Census-Income (KDD), (16) Chess (King-Rook vs. King), (17) Chess (King-Rook vs. King-Knight), (18) Chess (King-Rook vs. King-Pawn), (19) Congressional Voting Records, (20) Communities and Crime, (21) Contraceptive Method Choice, (22) Credit Approval, (23) Cylinder Bands, (24) Dermatology, (25) Dorothea, (26) Echocardiogram, (27) Ecoli, (28) Flags, (29) Forest Fires, (30)Gisette, (31) Glass Identification, (32) Haberman's Survival, (33) Hayes-Roth, (34) Heart Disease, (35) Hepatitise, (36) Horse Colic, (37) Housing, (38) Internet Advertisements, (39)Ionosphere, (40) Japanese Credit Screening, (41) Ionosphere, (42) Iris, (43) Letter Recognition, (44) Low Resolution Spectrometer, (45) Lung Cancer, (46) Magic Gamma Telescope, (47) MONK's Problems, (48) Mushroom, (49)Nursery, (50) Parkinsons, (51) Pima Indians Diabetes, (52) Pittsburgh Bridges, (53) Poker Hand, (54) Post-Operative Patient, (55) Soybean (large), (56) Spambase, (57) SPECHT Heart, (58)Statlog (shuttle), (59) Statlog (vehicle Silhouettes), (60) Teaching Assistant Evaluation, (61) Thyroid disease, (62)Tic-Tac-Toe Endgame, (63) University, (64) Wine, (65) Wine Quality, (66) Yeast and (67) Zoo.

F.1. Synthetic datasets

Feature interactions and feature redundancy are problematic in most data mining settings. The principal motivation behind generating synthetic datasets was to recreate these problems on large scale and perform experiments on controlled datsets.

Each dataset includes 10,000 instances each of 100 features. All features are continuous-valued. The response variable is a binary variable. The following steps were taken to generate these datasets.

Step 1: Specify different mean vectors and different covariance matrices for 90 features ($x_1,x_2,\ldots,x_{90}$) for the 200 different datasets. Since mean vectors and covariance matrices of no two datasets are the same, the joint distribution of features is different in each dataset.

Step 2: Generate 10,000 combinations of feature values for each dataset from its unique mean vector and covariance matrix.

Step 3: Create 10 new features ($x_{91},\ldots,x_{100}$)from the first 90 features, using the following equation:

$$x_{ij}=\alpha_0+\alpha_1 x_{i(j-15)}+\alpha_2 x_{i(j-22)}x_{i(j-27)}x_{i(j-50)}-\alpha_3 x_{i(j-66)}x_{i(j-74)}+\sigma_j R_{ij} \tag{21}$$

where $j=91,92,\ldots,99,100$; and $i=1,\ldots,10{,}000$; and $\sigma > 0x_{ij}$ represents the value of jth feature for the ith instance. $(\alpha_1,\alpha_2,\alpha_3)$ are model parameters.

To simulate interactions between features, we included two interaction terms. Interaction terms are formed by the multiplication of two or more explanatory variables. We included one two-way interaction term $(x_{i(x-66)}x_{i(x-74)})$, and one three-way interaction term $(x_{i(j-22)}x_{i(j-27)}x_{i(j-50)})$. R_{ij} is a normally distributed random number with mean 0 and standard deviation 1, σ_j denotes the standard error of the feature x_j.

Step 4: The probability of the event of interest for each instance was estimated by the following model. Only 6 features among 100 features were included in the model

$$P(Y) = 1/1+\exp(-Z), \quad Z = \beta_0+\beta_1x_{91}+\beta_2x_{94}+\beta_3x_{95}+\beta_4x_{97}-\beta_5x_{99}x_{100} \tag{22}$$

Table H1
The average classification accuracy of GRNN classifier in imputed datasets under different levels of missing rates (standard deviations in parentheses).

Algorithm	Rate of missing values								
	5%	10%	20%	30%	40%	50%	60%	70%	75%
GEMI	99 (3)	97 (3)	99 (5)	93 (2)	84 (7)	81 (9)	72 (13)	64 (7)	56 (14)
GESI	97 (3)	93 (4)	87 (7)	66 (14)	66 (14)	63 (7)	54 (7)	55 (8)	54 (21)
EM	99 (4)	82 (5)	72 (8)	66 (8)	67 (15)	65 (12)	55 (4)	45 (11)	47 (12)
GRNN MI	98 (4)	87 (9)	89 (3)	80 (9)	70 (17)	55 (11)	59 (13)	50 (12)	41 (16)
GRNN SI	98 (2)	84 (8)	67 (11)	67 (11)	56 (15)	56 (9)	55 (8)	51 (22)	54 (25)
HD MI	99 (3)	92 (5)	71 (12)	55 (14)	59 (12)	59 (8)	63 (10)	42 (18)	46 (16)
HD SI	98 (4)	90 (7)	64 (5)	62 (8)	66 (13)	54 (11)	51 (5)	50 (21)	57 (17)
HES MI	99 (4)	94 (5)	81 (2)	79 (9)	58 (14)	70 (11)	57 (5)	40 (19)	41 (23)
HES SI	99 (3)	84 (10)	97 (14)	65 (14)	57 (11)	53 (16)	58 (2)	54 (15)	49 (16)
HEW MI	99 (5)	82 (8)	93 (6)	81 (6)	72 (12)	55 (7)	67 (8)	51 (19)	55 (9)
HEW SI	99 (3)	99 (3)	90 (9)	70 (11)	55 (11)	51 (5)	67 (9)	43 (18)	45 (16)
HOS MI	97 (3)	90 (8)	71 (12)	95 (18)	71 (13)	63 (15)	58 (7)	51 (15)	46 (18)
HOS SI	98 (3)	86 (12)	60 (11)	62 (16)	63 (8)	57 (8)	53 (9)	43 (20)	58 (14)
HOW MI	99 (4)	93 (4)	83 (4)	90 (9)	75 (10)	68 (6)	58 (8)	70 (19)	48 (18)
HOW SI	99 (2)	87 (6)	89 (9)	93 (17)	45 (12)	58 (11)	57 (8)	55 (29)	56 (15)
KNN MI	99 (3)	93 (9)	83 (8)	59 (16)	73 (11)	54 (12)	55 (7)	56 (17)	51 (16)
KNN SI	99 (3)	81 (6)	61 (14)	60 (3)	60 (11)	53 (9)	71 (8)	47 (12)	56 (9)
MCMC	98 (3)	98 (4)	83 (3)	83 (10)	72 (12)	54 (15)	53 (5)	45 (14)	50 (15)
MLP MI	98 (3)	97 (5)	78 (7)	70 (11)	63 (13)	58 (9)	62 (9)	46 (14)	43 (12)
MLP SI	99 (4)	82 (7)	72 (11)	56 (13)	59 (17)	59 (9)	53 (13)	47 (10)	55 (15)
MS	99 (3)	82 (5)	64 (7)	66 (10)	57 (14)	59 (9)	64 (9)	51 (9)	47 (10)
RBF MI	97 (3)	83 (10)	80 (9)	69 (10)	49 (16)	54 (10)	51 (9)	52 (17)	49 (24)
RBF SI	98 (3)	85 (9)	70 (6)	72 (15)	54 (8)	49 (17)	55 (7)	37 (13)	49 (20)
WKNN MI	99 (4)	86 (8)	78 (3)	71 (12)	67 (22)	56 (9)	52 (4)	48 (17)	46 (19)
WKNN SI	97 (2)	82 (8)	76 (10)	76 (6)	66 (17)	61 (10)	64 (7)	53 (12)	41 (17)
ZI	98 (4)	80 (12)	54 (18)	60 (9)	64 (10)	57 (8)	55 (6)	45 (18)	60 (17)

Table H2
The average classification accuracy of LR classifier in imputed datasets under different levels of missing rates (standard deviations in parentheses).

Algorithms	Rate of missing values								
	5%	10%	20%	30%	40%	50%	60%	70%	75%
GEMI	98 (4)	**96 (5)**	**93 (6)**	**85 (8)**	**74 (10)**	**64 (21)**	**59 (14)**	**52 (11)**	**56 (18)**
GESI	94 (3)	85 (9)	75 (8)	63 (17)	**66 (24)**	54 (8)	52 (11)	**55 (10)**	39 (12)
EM	98 (4)	77 (7)	62 (10)	60 (7)	51 (29)	51 (17)	49 (5)	44 (16)	44 (17)
GRNN MI	98 (4)	82 (7)	76 (11)	69 (10)	48 (21)	55 (12)	51 (11)	44 (12)	45 (16)
GRNN SI	98 (5)	81 (11)	62 (14)	59 (19)	55 (17)	49 (8)	51 (9)	33 (12)	46 (14)
HD MI	98 (4)	79 (7)	64 (12)	52 (20)	55 (15)	**58 (11)**	50 (10)	38 (18)	44 (18)
HD SI	98 (4)	88 (8)	60 (7)	50 (23)	59 (16)	53 (12)	47 (7)	38 (15)	52 (24)
HES MI	98 (4)	83 (7)	71 (14)	63 (14)	47 (21)	53 (12)	50 (9)	41 (21)	39 (15)
HES SI	98 (4)	77 (9)	66 (14)	56 (15)	47 (22)	52 (18)	55 (9)	53 (17)	42 (11)
HEW MI	98 (5)	74 (5)	**87 (7)**	73 (14)	60 (19)	52 (11)	53 (16)	39 (18)	38 (19)
HEW SI	98 (6)	72 (7)	69 (15)	54 (12)	55 (14)	50 (10)	43 (9)	43 (19)	43 (21)
HOS MI	98 (4)	75 (10)	67 (19)	70 (20)	53 (22)	49 (17)	**53 (8)**	45 (16)	38 (19)
HOS SI	98 (4)	77 (16)	51 (19)	47 (18)	55 (23)	51 (13)	50 (10)	39 (20)	46 (14)
HOW MI	98 (4)	85 (10)	**82 (11)**	**85 (10)**	**74 (11)**	**58 (12)**	52 (11)	36 (13)	47 (18)
HOW SI	98 (4)	80 (6)	71 (11)	64 (18)	45 (22)	46 (19)	43 (9)	40 (14)	43 (15)
KNN MI	98 (3)	82 (10)	60 (14)	57 (18)	55 (18)	54 (14)	48 (12)	36 (16)	40(20)
KNN SI	98 (4)	72 (19)	57 (18)	47 (21)	51 (20)	47 (10)	51 (10)	43 (16)	47 (23)
MCMC	98 (4)	**95 (7)**	73 (6)	76 (11)	57 (15)	49 (14)	49 (8)	41 (20)	49 (18)
MLP MI	98 (3)	75 (11)	71 (11)	62 (15)	59 (18)	57 (14)	47 (10)	42 (16)	43 (22)
MLP SI	98 (5)	79 (10)	47 (11)	51 (20)	55 (19)	46 (13)	50 (16)	45 (14)	42 (15)
MS	98 (4)	73 (10)	50 (8)	47 (26)	55 (32)	49 (7)	50 (13)	44 (17)	39 (18)
RBF MI	98 (4)	80 (13)	64 (13)	65 (11)	46 (24)	46 (12)	47 (12)	37 (11)	34 (18)
RBF SI	98 (3)	80 (11)	56 (7)	37 (22)	54 (14)	47 (17)	50 (9)	39 (15)	45 (16)
WKNN MI	98 (4)	79 (9)	65 (9)	53 (24)	45 (21)	50 (12)	52 (12)	43 (12)	41 (16)
WKNN SI	98 (4)	79 (13)	53 (14)	41 (29)	51 (18)	50 (11)	52 (9)	49 (21)	40 (18)
ZI	98 (5)	70 (18)	51 (13)	47 (25)	53 (14)	44 (9)	45 (8)	40 (18)	47 (17)

Table H3
The average classification accuracy of MLP classifier in imputed datasets under different levels of missing rates (standard deviations in parentheses).

Algorithms	Rate of missing values								
	5%	10%	20%	30%	40%	50%	60%	70%	75%
GEMI	94 (5)	**94 (5)**	**84 (6)**	**77 (10)**	**71 (10)**	**63 (22)**	46 (23)	49 (17)	19 (13)
GESI	90 (5)	77 (4)	63 (10)	56 (15)	32 (13)	49 (14)	48 (11)	48 (15)	27 (12)
EM	95 (5)	65 (7)	50 (20)	53 (20)	38 (15)	43 (19)	42 (17)	40 (11)	37 (10)
GRNN MI	96 (5)	67 (14)	48 (20)	54 (12)	36 (19)	46 (19)	39 (10)	39 (17)	42 (16)
GRNN SI	94 (5)	74 (15)	62 (13)	45 (27)	**54 (19)**	44 (10)	44 (9)	32 (9)	39 (7)
HD MI	96 (4)	72 (8)	51 (14)	49 (26)	34 (21)	46 (15)	37 (3)	37 (17)	40 (16)
HD SI	95 (4)	63 (10)	59 (13)	42 (29)	26 (17)	45 (14)	44 (12)	28 (14)	30 (13)
HES MI	99 (4)	67 (14)	62 (18)	34 (20)	41 (34)	53 (27)	43 (17)	36 (18)	34 (12)
HES SI	93 (5)	77 (11)	23 (18)	41 (16)	40 (27)	44 (20)	42 (13)	22 (12)	38 (12)
HEW MI	97 (5)	**89 (7)**	39 (10)	45 (18)	**51 (20)**	42 (13)	26 (12)	33 (10)	29 (13)
HEW SI	95 (6)	71 (16)	45 (13)	54 (13)	47 (22)	47 (12)	39 (11)	40 (15)	37 (16)
HOS MI	94 (5)	71 (13)	66 (21)	34 (19)	32 (17)	42 (16)	45 (11)	35 (14)	32 (14)
HOS SI	97 (5)	57 (18)	48 (18)	46 (19)	**52 (24)**	49 (17)	50 (12)	37 (16)	37 (18)
HOW MI	95 (6)	79 (11)	75 (12)	50 (16)	**53 (17)**	48 (19)	39 (13)	22 (12)	28 (15)
HOW SI	92 (5)	**80 (11)**	52 (21)	31 (22)	39 (14)	38 (16)	42 (9)	26 (12)	38 (14)
KNN MI	96 (5)	63 (15)	41 (16)	47 (23)	39 (18)	51 (15)	42 (14)	27 (11)	27 (13)
KNN SI	94 (6)	65 (21)	50 (22)	45 (42)	**53 (22)**	45 (12)	23 (7)	42 (17)	25 (12)
MCMC	96 (6)	69 (9)	**85 (13)**	**61 (12)**	**56 (17)**	41 (19)	49 (13)	32 (14)	28 (14)
MLP MI	94 (4)	65 (10)	47 (15)	58 (18)	50 (18)	39 (12)	46 (11)	42 (21)	37 (14)
MLP SI	95 (6)	61 (15)	48 (10)	47 (22)	49 (19)	38 (14)	23 (12)	40 (11)	29 (11)
MS	97 (4)	65 (11)	49 (18)	40 (31)	39 (17)	48 (12)	33 (12)	35 (15)	36 (16)
RBF MI	96 (5)	74 (14)	62 (14)	59 (11)	39 (21)	46 (16)	46 (12)	31 (14)	33 (16)
RBF SI	95 (5)	68 (14)	55 (20)	37 (15)	45 (18)	49 (19)	48 (9)	39 (17)	36 (16)
WKNN MI	96 (5)	70 (15)	55 (21)	55 (27)	39 (17)	48 (16)	46 (11)	39 (15)	36 (15)
WKNN SI	95 (5)	64 (17)	48 (14)	38 (32)	49 (20)	42 (15)	40 (10)	37 (16)	37 (17)
ZI	94 (6)	56 (16)	44 (24)	38 (2)	48 (21)	38 (11)	43 (10)	37 (15)	37 (12)

Table H4
Pairwise comparisons among imputation algorithms in terms of interval estimation of missing data.

Rank	Algorithm	Significantly outperformed algorithms
With about 10% missing values		
1	GEMI	(1)MCMC, (2) HOW MI, (3) HEW MI, (4) GRNN MI, (5) HD MI, (6) WKNN MI, (7) HES MI, (8) RBF MI, (9) MLP MI, (10) HOS MI, (11) KNN MI
2	MCMC	(1) GRNN MI, (2) HD MI, (3) WKNN MI, (4) HES MI, (5) RBF MI, (6) MLP MI, (7) HOS MI, (8) KNN MI
3	HOW MI, HEW MI, GRNN MI, HD MI, WKNN MI	(1) HES MI, (2) RBF_MI, (3) MLP MI, (4) HOS MI, (5) KNN MI
4	HES MI	(1) RBF MI, (2) MLP MI, (3) HOS MI, (4) KNN MI
5	RBF MI	(1) MLP MI, (2) HOS MI, (3) KNN MI
6	MLP MI	(1) HOS MI, (2) KNN MI
7	HOS MI	(1) KNN MI
8	KNN MI	0
With about 20% missing values		
1	GEMI	(1) HOW MI, (2) HEW MI, (3) GRNN MI, (4) HES MI, (5) RBF MI, (6) WKNN MI, (7) HD MI, (8) KNN MI, (9) MLP MI, (10) HOS MI
2	MCMC, HOW MI	(1) HEW MI, (2) GRNN MI, (3) HES MI, (4) RBF MI, (5) WKNN MI, (6) HD MI, (7) KNN MI, (8) MLP MI, (9) HOS MI
3	HEW MI	(1) GRNN MI, (2) HES MI, (3) RBF MI, (4) WKNN MI, (5) HD MI, (6) KNN MI, (7) MLP MI, (8) HOS MI
4	GRNN MI	(1) HES MI, (2) RBF MI, (3) WKNN MI, (4) HD MI, (5) KNN MI, (6) MLP MI, (7) HOS MI
5	HES MI, RBF MI	(1) WKNN MI, (2) HD MI, (3) KNN MI, (4) MLP MI, (5) HOS MI
6	WKNN MI	(1) HD MI, (2) KNN MI, (3) MLP MI, (4) HOS MI
7	HD MI	(1) KNN MI, (2) MLP MI, (3) HOS MI
8	KNN MI	(1) MLP MI, (2) HOS MI
9	MLP MI	(1) HOS MI
10	HOS MI	0
With about 30% missing values		
1	GEMI	(1) MCMC, (2) HOW MI, (3) HEW MI, (4) HOS MI, (5)HES MI, (6) GRNN MI, (7) HD MI, (8) RBF MI, (9) MLP MI, (10)KNN MI, (11) WKNN MI
2	MCMC	(1) HOS MI, (2)HES MI, (3) GRNN MI, (4) HD MI, (5) RBF MI, (6) MLP MI, (7)KNN MI, (8) WKNN MI
3	HOW MI, HEW MI	(1) GRNN MI, (2) HD MI, (3) RBF MI, (4) MLP MI, (5)KNN MI, (6) WKNN MI
4	HOS MI	(1) RBF MI, (2) MLP MI, (3)KNN MI, (4) WKNN MI
5	HES MI, GRNN MI, HD MI, RBF MI	(1) MLP MI, (2)KNN MI, (3) WKNN MI
6	MLP MI	(1)KNN MI, (2) WKNN MI
7	KNN MI	WKNN MI
8	WKNN MI	0

Table H4 (*continued*)

Rank	Algorithm	Significantly outperformed algorithms
With about 40% missing values		
1	GEMI	(1) MCMC, (2) HEW MI, (3) HD MI, (4) HES MI, (5) MLP MI, (6) KNN MI, (7) WKNN MI, (8) HOS MI, (9) RBF MI
2	HOW MI, GRNN MI	(1) MLP MI, (2) KNN MI, (3) WKNN MI, (4) HOS MI, (5) RBF MI
3	MCMC,HEW MI, HD MI	(1) KNN MI, (2) WKNN MI, (3) HOS MI, (4) RBF MI
4	HES MI, MLP MI	(1) WKNN MI, (2) HOS MI, (3) RBF MI
5	KNN MI	(1) HOS MI, (2) RBF MI
6	WKNN MI	(1) RBF MI
7	RBF MI	0
With about 50% missing values		
1	GEMI	(1)HOW MI, (2) HES MI, (3) RBF MI, (4) MCMC, (5) HEW MI, (6) GRNN MI, (7) HD MI, (8) MLP MI, (9) KNN MI, (10) HOS MI, (11) WKNN MI
2	HOW MI	(1) MCMC, (2) HEW MI, (3) GRNN MI, (4) HD MI, (5) MLP MI, (6) KNN MI, (7) HOS MI, (8) WKNN MI
3	HES MI, RBF MI	(1) MLP MI, (2) KNN MI, (3) HOS MI, (4) WKNN MI
4	MCMC, HEW MI, GRNN MI, HD MI	(1)HOS MI, (2) WKNN MI
5	KNN MI, HOS MI	WKNN MI
6	WKNN MI	0
With about 60% missing values		
1	GEMI	(1) HOW MI, (2) HEW MI, (3) HES MI, (4) HD MI, (5) KNN MI, (6) GRNN MI, (7) HOS MI, (8) MCMC, (9) RBF MI, (10) MLP MI, (11) WKNN MI
2	HOW MI	(1) HOS MI, (2) MCMC, (3) RBF MI, (4) MLP MI, (5) WKNN MI
3	HEW MI, HES MI, HD MI, KNN MI	(1) MCMC, (2) RBF MI, (3) MLP MI, (4) WKNN MI
4	GRNN MI, HOS MI	(1) MLP MI, (2) WKNN MI
5	MCMC, RBF MI, MLP MI	(1) WKNN MI
6	WKNN MI	0
With about 70% missing values		
1	GEMI	(1) HOW MI, (2) HEW MI, (3) HES MI, (4) HD MI, (5) KNN MI, (6) GRNN MI, (7) HOS MI, (8) MCMC, (9) RBF MI, (10) MLP MI, (11) WKNN MI
2	HOW MI, HEW MI, HES MI, HD MI, KNN MI, GRNN MI, HOS MI, MCMC, RBF MI, MLP MI, WKNN MI	0

Table H5
Pairwise comparisons among imputation algorithms in terms of the accuracy of estimating missing values.

Rank	Algorithm	Significantly outperformed algorithms
With about 10% missing values		
1	GESI	(1) HOW SI, (2)HD SI, (3) GRNN SI, (4) EM, (5) WKNN SI, (6) HEW SI, (7) KNN SI, (8)ZI, (9) HES SI, (10)MLP SI, (11) HOS SI, (12) MS, (13) GEMI, (14) RBF SI, (15) MCMC, (16) HOW MI, (17) WKNN MI, (18)HEW MI, (19)MLP MI, (20) KNN MI, (21)GRNN MI, (22) RBF MI, (23) HOS MI, (24) HD MI, (25) HES MI
2	HOW SI, HD SI, GRNN SI, EM, WKNN SI	(1)MLP SI, (2) HOS SI, (3) MS, (4) GEMI, (5) RBF SI, (6) MCMC, (7) HOW MI, (8) WKNN MI, (9)HEW MI, (10)MLP MI, (11) KNN MI, (12)GRNN MI, (13) RBF MI, (14) HOS MI, (15) HD MI, (16) HES MI
3	HEW SI	(1) HOS SI, (2) MS, (3) GEMI, (4) RBF SI, (5) MCMC, (6) HOW MI, (7) WKNN MI, (8)HEW MI, (9)MLP MI, (10) KNN MI, (11)GRNN MI, (12) RBF MI, (13) HOS MI, (14) HD MI, (15) HES MI
4	KNN SI, ZI	(1) GEMI, (2) RBF SI, (3) MCMC, (4) HOW MI, (5) WKNN MI, (6)HEW MI, (7)MLP MI, (8) KNN MI, (9)GRNN MI, (10) RBF MI, (11) HOS MI, (12) HD MI, (13) HES MI
5	HES SI, MLP SI, HOS SI, MS	(1) RBF SI, (2) MCMC, (3) HOW MI, (4) WKNN MI, (5)HEW MI, (6)MLP MI, (7) KNN MI, (8)GRNN MI, (9) RBF MI, (10) HOS MI, (11) HD MI, (12) HES MI
6	GEMI, RBF SI	(1) MCMC, (2) HOW MI, (3) WKNN MI, (4)HEW MI, (5)MLP MI, (6) KNN MI, (7)GRNN MI, (8) RBF MI, (9) HOS MI, (10) HD MI, (11) HES MI
7	MCMC	(1) HOW MI, (2) WKNN MI, (3)HEW MI, (4)MLP MI, (5) KNN MI, (6)GRNN MI, (7) RBF MI, (8) HOS MI, (9) HD MI, (10) HES MI
8	HOW MI	(1) WKNN MI, (2)HET_MI2, (3)MLP MI, (4) KNN MI, (5)GRNN MI, (6) RBF MI, (7) HOM_MI1, (8) HD MI, (9) HET_MI1
9	WKNN MI	(1)HEW MI, (2)MLP MI, (3) KNN MI, (4)GRNN MI, (5) RBF MI, (6) HOS MI, (7) HD MI, (8) HES MI
10	HEW MI	(1)MLP MI, (2) KNN MI, (3)GRNN MI, (4) RBF MI, (5) HOM_MI1, (6) HD MI, (7) HET_MI1
11	MLP MI	(1) KNN MI, (2)GRNN MI, (3) RBF MI, (4) HOS MI, (5) HD MI, (6) HES MI
12	KNN MI	(1)GRNN MI, (2) RBF MI, (3) HOS MI, (4) HD MI, (5) HES MI
13	GRNN MI	(1) RBF MI, (2) HOS MI, (3) HD MI, (4) HES MI
14	RBF MI	(1) HOS MI, (2) HD MI, (3) HES MI
15	HOS MI	(1) HD MI, (2) HES MI
16	HD MI	HES MI
17	HES MI	0

Table H5 (*continued*)

Rank	Algorithm	Significantly outperformed algorithms
With about 20% missing values		
1	GESI	(1) EM, (2) MLP SI, (3) KNN SI, (4) HOS SI, (5) MS, (6) GEMI, (7) HD SI, (8) WKNN SI, (9) HOW MI, (10) MCMC, (11) HES SI, (12) ZI, (13) RBF MI, (14) HEW MI, (15) KNN MI, (16) GRNN MI, (17) WKNN MI, (18) HES MI, (19) MLP MI, (20) HOS MI, (21) HD MI
2	GRNN SI	(1) HD SI, (2) WKNN SI, (3) HOW MI, (4) MCMC, (5) HES SI, (6) ZI, (7) RBF MI, (8) HEW MI, (9) KNN MI, (10) GRNN MI, (11) WKNN MI, (12) HES MI, (13) MLP MI, (14) HOS MI, (15) HD MI
3	HOW SI, HEW SI, RBF SI, EM, MLP SI, KNN SI, HOS SI, MS	(1) WKNN SI, (2) HOW MI, (3) MCMC, (4) HES SI, (5) ZI, (6) RBF MI, (7) HEW MI, (8) KNN MI, (9) GRNN MI, (10) WKNN MI, (11) HET_MI1, (12) MLP MI, (13) HOS MI, (14) HD MI
4	GEMI, HD SI, WKNN SI	(1) HOM_MI2, (2) MCMC, (3) HES SI, (4) ZI, (5) RBF MI, (6) HEW MI, (7) KNN MI, (8) GRNN MI, (9) WKNN MI, (10) HES MI, (11) MLP MI, (12) HOS MI, (13) HD MI
5	HOW MI	(1) MCMC, (2) HES SI, (3) ZI, (4) RBF MI, (5) HEW MI, (6) KNN MI, (7) GRNN MI, (8) WKNN MI, (9) HES MI, (10) MLP MI, (11) HOS MI, (12) HD MI
6	MCMC	(1) HES SI, (2) ZI, (3) RBF MI, (4) HEW MI, (5) KNN MI, (6) GRNN MI, (7) WKNN MI, (8) HES MI, (9) MLP MI, (10) HOS MI, (11) HD MI
7	HES SI	(1) ZI, (2) RBF MI, (3) HEW MI, (4) KNN MI, (5) GRNN MI, (6) WKNN MI, (7) HES MI, (8) MLP MI, (9) HOS MI, (10) HD MI
8	ZI	(1) RBF MI, (2) HEW MI, (3) KNN MI, (4) GRNN MI, (5) WKNN MI, (6) HES MI, (7) MLP MI, (8) HOS MI, (9) HD MI
9	RBF MI	(1) HEW MI, (2) KNN MI, (3) GRNN MI, (4) WKNN MI, (5) HES MI, (6) MLP MI, (7) HOS MI, (8) HD MI
10	HEW MI	(1) KNN MI, (2) GRNN MI, (3) WKNN MI, (4) HES MI, (5) MLP MI, (6) HOS MI, (7) HD MI
11	KNN MI	(1) GRNN MI, (2) WKNN MI, (3) HES MI, (4) MLP MI, (5) HOS MI, (6) HD MI
12	GRNN MI	(1) WKNN MI, (2) HES MI, (3) MLP MI, (4) HOS MI1, (5) HD MI
13	WKNN MI	(1) HES MI, (2) MLP MI, (3) HOS MI, (4) HD MI
14	HES MI	(1) MLP MI, (2) HOS MI, (3) HD MI
15	MLP MI	(1) HOS MI, (2) HD MI
16	HOS MI	(1) HD MI
17	HD MI	0
With about 30% missing values		
1	GESI	(1) HES SI, (2) GRNN SI, (3) EM, (4) MLP SI, (5) KNN SI, (6) GEMI, (7) RBF SI, (8) MS, (9) HES SI, (10) WKNN SI, (11) HOS SI, (12) RBF MI, (13) KNN MI, (14) MCMC, (15) WKNN MI, (16) HOW MI, (17) HOW SI, (18) HD SI, (19) HEW MI, (20) GRNN MI, (21) HD MI, (22) HES MI, (23) ZI, (24) MLP MI, (25) HOS MI
2	HEW SI, GRNN SI, EM, MLP SI, KNN SI,	(1) KNN MI, (2) MCMC, (3) WKNN MI, (4) HOW MI, (5) HOW SI, (6) HD SI, (7) HEW MI, (8) GRNN MI, (9) HD MI, (10) HES MI, (11) ZI, (12) MLP MI, (13) HOS MI
3	GEMI, RBF SI, MS	(1) MCMC, (2) WKNN MI, (3) HOW MI, (4) HOW SI, (5) HD SI, (6) HEW MI, (7) GRNN MI, (8) HD MI, (9) HES MI, (10) ZI, (11) MLP MI, (12) HOS MI
4	HES SI, WKNN SI, HOS SI	(1) WKNN MI, (2) HOW MI, (3) HOW SI, (4) HD SI, (5) HEW MI, (6) GRNN MI, (7) HD MI, (8) HEW MI, (9) ZI, (10) MLP MI, (11) HOS MI
5	RBF MI	(1) HOW MI, (2) HOW SI, (3) HD SI, (4) HEW MI, (5) GRNN MI, (6) HD MI, (7) HES MI, (8) ZI, (9) MLP MI, (10) HOS MI
6	KNN MI	(1) HOW SI, (2) HD SI, (3) HEW MI, (4) GRNN MI, (5) HD MI, (6) HES MI, (7) ZI, (8) MLP MI, (9) HOS MI
7	MCMC, WKNN MI	(1) HD SI, (2) HEW MI, (3) GRNN MI, (4) HD MI, (5) HES MI, (6) ZI, (7) MLP MI, (8) HOS MI
8	HOW MI, HOW SI, HD SI	(1) HEW MI, (2) GRNN MI, (3) HD MI, (4) HES MI, (5) ZI, (6) MLP MI, (7) HOS MI
9	HEW MI	(1) GRNN MI, (2) HD MI, (3) HES MI, (4) ZI, (5) MLP MI, (6) HOS MI
10	GRNN MI	(1) HD MI, (2) HES MI, (3) ZI, (4) MLP MI, (5) HOS MI
11	HD MI	(1) HES MI, (2) ZI, (3) MLP MI, (4) HOS MI
12	HES MI	(1) ZI, (2) MLP MI, (3) HOS MI
13	ZI	(1) MLP MI, (2) HOS MI
14	MLP MI	(1) HOS MI
15	HOS MI	0
With about 40% missing values		
1	GESI	(1) GRNN SI, (2) MLP SI, (3) WKNN SI, (4)MCMC, (5) HD SI, (6) RBF MI, (7) KNN MI, (8) HES SI, (9) HOS SI, (10) GRNN MI, (11) HOW MI, (12) HEW MI, (13) KNN SI, (14) HES MI, (15) MS, (16) ZI, (17) HOS MI, (18) WKNN MI, (19) HD MI, (20) MLP MI
2	RBF SI	(1) HD SI, (2) RBF MI, (3) KNN MI, (4) HES SI, (5) HOS SI, (6) GRNN MI, (7) HOW MI, (8) HEW MI, (9) KNN SI, (10) HES MI, (11) MS, (12) ZI, (13) HOS MI, (14) WKNN MI, (15) HD MI, (16) MLP MI
3	EM	(1) KNN MI, (2) HES SI, (3) HOS SI, (4) GRNN MI, (5) HOW MI, (6) HEW MI, (7) KNN SI, (8) HES MI, (9) MS, (10) ZI, (11) HOS MI, (12) WKNN MI, (13) HD MI, (14) MLP MI
4	GEMI, HEW SI	(1) HES SI, (2) HOS SI, (3) GRNN MI, (4) HOW MI, (5) HEW MI, (6) KNN SI, (7) HES MI, (8) MS, (9) ZI, (10) HOS MI, (11) WKNN MI, (12) HD MI, (13) MLP MI
5	HOW SI, GRNN SI, MLP SI, WKNN SI	

Table H5 (*continued*)

Rank	Algorithm	Significantly outperformed algorithms
		(1) HOS SI, (2) GRNN MI, (3) HOW MI, (4) HEW MI, (5) KNN SI, (6) HES MI, (7) MS, (8) ZI, (9) HOS MI, (10) WKNN MI, (11) HD MI, (12) MLP MI
6	MCMC, HD SI	(1) HEW MI, (2) KNN SI, (3) HES MI, (4) MS, (5) ZI, (6) HOS MI, (7) WKNN MI, (8) HD MI, (9) MLP MI
7	RBF MI, KNN MI	(1) KNN SI, (2) HES MI, (3) MS, (4) ZI, (5) HOS MI, (6) WKNN MI, (7) HD MI, (8) MLP MI
8	HES SI, HOS SI	(1) HES MI, (2) MS, (3) ZI, (4) HOS MI, (5) WKNN MI, (6) HD MI, (7) MLP MI
9	GRNN MI	(1) MS, (2) ZI, (3) HOS MI, (4) WKNN MI, (5) HD MI, (6) MLP MI
10	HOW MI, HEW MI, KNN SI	(1) ZI, (2) HOS MI, (3) WKNN MI, (4) HD MI, (5) MLP MI
11	HES MI, MS, ZI	(1) HOS MI, (2) WKNN MI, (3) HD MI, (4) MLP MI
12	HOS MI	(1) WKNN MI, (2) HD MI, (3) MLP MI
13	WKNN MI	(1) HD MI, (2) MLP MI
14	HD MI	(1) MLP MI
15	MLP MI	0
With about 50% missing values		
1	GESI	(1) MLP SI, (2) GEMI, (3) GRNN SI, (4) HES SI, (5) MS, (6) MCMC, (7) HEW MI, (8) HES MI, (9) HOS MI, (10) HOW MI, (11) GRNN MI, (12) ZI, (13) HD SI, (14) MLP MI, (15) HOS SI, (16) HOW SI, (17)WKNN MI, (18) KNN
2	EM	(1) MCMC, (2) HEW MI, (3) HES MI, (4) HOS MI, (5) HOW MI, (6) GRNN MI, (7) ZI, (8) HD SI, (9) MLP MI, (10) HOS SI, (11) HOW SI, (12)WKNN MI, (13) KNN SI, (14) HD MI, (15) RBF MI, (16) KNN MI
3	RBF SI	(1) HOS MI, (2) HOW MI, (3) GRNN MI, (4) ZI, (5) HD SI, (6) MLP MI, (7) HOS SI, (8) HOW SI, (9)WKNN MI, (10) KNN SI, (11) HD MI, (12) RBF MI, (13) KNN MI
4	HEW SI	(1) HOW MI, (2) GRNN MI, (3) ZI, (4) HD SI, (5) MLP MI, (6) HOS SI, (7) HOW SI, (8)WKNN MI, (9) KNN SI, (10) HD MI, (11) RBF MI, (12) KNN MI
5	WKNN SI	(1) ZI, (2) HD SI, (3) MLP MI, (4) HOS SI, (5) HOW SI, (6)WKNN MI, (7) KNN SI, (8) HD MI, (9) RBF MI, (10) KNN MI
6	MLP SI	(1) HD SI, (2) MLP MI, (3) HOS SI, (4) HOW SI, (5)WKNN MI, (6) KNN SI, (7) HD MI, (8) RBF MI, (9) KNN MI
7	GEMI, GRNN SI	(1) MLP MI, (2) HOS SI, (3) HOW SI, (4)WKNN MI, (5) KNN SI, (6) HD MI, (7) RBF MI, (8) KNN MI
8	HES SI, MS	(1) HOS SI, (2) HOW SI, (3)WKNN MI, (4) KNN SI, (5) HD MI, (6) RBF MI, (7) KNN MI
9	MCMC, HEW MI, HES MI, HOS MI	(1) HOW SI, (2)WKNN MI, (3) KNN SI, (4) HD MI, (5) RBF MI, (6) KNN MI
10	HOW MI, GRNN MI, ZI	(1)WKNN MI, (2) KNN SI, (3) HD MI, (4) RBF MI, (5) KNN MI
11	HD SI, MLP MI, HOS SI	(1) KNN SI, (2) HD MI, (3) RBF MI, (4) KNN MI
12	HOW SI, WKNN MI, KNN SI	(1) HD MI, (2) RBF MI, (3) KNN MI
13	HD MI	(1) RBF MI, (2) KNN MI
14	RBF MI	KNN MI
15	KNN MI	0
With about 60% missing values		
1	GESI	(1) HD SI, (2) MLP SI, (3) GEMI, (4) GRNN SI, (5) HES SI, (6) WKNN SI, (7) MS, (8) MCMC, (9) HEW MI, (10) HOS MI, (11) HOW MI, (12) RBF MI, (13) ZI, (14) MLP MI, (15) WKNN MI, (16) HOS SI, (17) HOW SI, (18) KNN SI, (19) HES MI, (20) KNN MI, (21) HD MI
2	EM	(1) WKNN SI, (2) MS, (3) MCMC, (4) HEW MI, (5) HOS MI, (6) HOW MI, (7) RBF MI, (8) ZI, (9) MLP MI, (10) WKNN MI, (11) HOS SI, (12) HOW SI, (13) KNN SI, (14) HES MI, (15) KNN MI, (16) HD MI
3	RBF SI	(1) HEW MI, (2) HOS MI, (3) HOW MI, (4) RBF MI, (5) ZI, (6) MLP MI, (7) WKNN MI, (8) HOS SI, (9) HOW SI, (10) KNN SI, (11) HES MI, (12) KNN MI, (13) HD MI
4	HEW SI	(1) HOS MI, (2) HOW MI, (3) RBF MI, (4) ZI, (5) MLP MI, (6) WKNN MI, (7) HOS SI, (8) HOW SI, (9) KNN SI, (10) HES MI, (11) KNN MI, (12) HD MI
5	GRNN MI, HD SI	(1) RBF MI, (2) ZI, (3) MLP MI, (4) WKNN MI, (5) HOS SI, (6) HOW SI, (7) KNN SI, (8) HES MI, (9) KNN MI, (10) HD MI
6	MLP SI	(1) ZI, (2) MLP MI, (3) WKNN MI, (4) HOS SI, (5) HOW SI, (6) KNN SI, (7) HES MI, (8) KNN MI, (9) HD MI
7	GEMI, GRNN SI	(1) MLP MI, (2) WKNN MI, (3) HOS SI, (4) HOW SI, (5) KNN SI, (6) HES MI, (7) KNN MI, (8) HD MI
8	HES SI, WKNN SI, MS	(1) WKNN MI, (2) HOS SI, (3) HOW SI, (4) KNN SI, (5) HES MI, (6) KNN MI, (7) HD MI
9	MCMC, HEW MI, HOS MI	(1) HOS SI, (2) HOW SI, (3) KNN SI, (4) HES MI, (5) KNN MI, (6) HD MI
10	HOW MI, RBF MI, ZI	(1) HOS SI, (2) KNN SI, (3) HES MI, (4) KNN MI, (5) HD MI
11	MLP MI, WKNN MI, HOS SI	(1) KNN SI, (2) HES MI, (3) KNN MI, (4) HD MI
12	HOW SI, KNN SI	(1) HES MI, (2) KNN MI, (3) HD MI
13	HES MI	(1) KNN MI, (2) HD MI
14	KNN MI	(1) HD MI
15	HD MI	0
With about 70% missing values		
1	GESI, GEMI	(1) HES SI, (2) GRNN SI, (3) EM, (4) MLP SI, (5) KNN SI, (6) RBF SI, (7) MS, (8) HEW SI, (9) WKNN SI, (10) HOS SI, (11) RBF MI, (12) KNN MI, (13) MCMC, (14) WKNN MI, (15) HOW MI, (16) HOW SI, (17) HD SI, (18) HEW MI, (19) GRNN MI, (20) HD MI, (21) HES MI, (22) ZI, (23) MLP MI, (24) HOS MI
2	HES SI, GRNN SI, EM, MLP SI, KNN SI, RBF SI, MS, HEW SI, WKNN SI, HOS SI, RBF MI, KNN MI, MCMC, WKNN MI, HOW MI, HOS_SI, HD SI, HEW MI, GRNN MI, HD MI, HES MI, ZI, MLP MI, HOS MI	0

Table H6
Comparison of the computational cost of imputation algorithms (standard deviations in parentheses).

	Computation time in hours								
Rate of missing values	5%	10%	20%	30%	40%	50%	60%	70%	75%
GEMI	39 (9)	48 (10)	66 (13)	93 (19)	110 (22)	147 (26)	94 (33)	58 (24)	56 (25)
GESI	25 (7)	28 (7)	35 (14)	46 (17)	52 (21)	61 (25)	45 (31)	45 (24)	48 (22)
EM	27 (10)	29 (12)	37 (13)	51 (20)	57 (24)	70 (28)	47 (34)	44 (26)	43 (24)
GRNN MI	36 (7)	43 (10)	57 (13)	77 (19)	92 (22)	110(27)	37 (34)	36 (25)	37 (22)
GRNN SI	24 (8)	25 (9)	30 (13)	37 (17)	40 (21)	43 (26)	35 (30)	33 (24)	31 (23)
HD MI	35 (7)	41 (9)	53 (14)	67 (18)	82 (21)	96 (26)	27 (16)	32 (24)	30 (22)
HD SI	23 (7)	23 (10)	26 (15)	29 (17)	34 (22)	36 (25)	24 (12)	24 (13)	27 (22)
HES MI	44 (9)	52 (15)	79 (19)	105 (22)	136 (24)	162 (30)	59 (36)	53 (29)	55 (24)
HES SI	30 (10)	33 (11)	42 (13)	49 (20)	56 (23)	59 (22)	35 (30)	47 (25)	47 (23)
HEW MI	47 (11)	56 (17)	75 (20)	122 (24)	148 (27)	178 (31)	62 (40)	60 (29)	62 (28)
HEW SI	30 (8)	39 (13)	46 (15)	72 (19)	77 (23)	89 (28)	48 (31)	39 (26)	40 (27)
HOS MI	38 (9)	45 (16)	62 (21)	95 (23)	110 (25)	140 (29)	43 (31)	45 (28)	44 (28)
HOS SI	26 (7)	27 (12)	31 (17)	44 (19)	48 (23)	56 (27)	32 (19)	31 (25)	32 (19)
HOW MI	43 (11)	50 (17)	70 (18)	102 (24)	129 (26)	152 (31)	46 (32)	48 (29)	48 (26)
HOW SI	28 (11)	31 (10)	38 (15)	63 (18)	69 (25)	80 (28)	34 (16)	30 (14)	31 (20)
KNN MI	34 (8)	51 (13)	54 (17)	68 (19)	82 (21)	100 (25)	41 (30)	37 (24)	35 (22)
KNN SI	22 (7)	23 (10)	27 (19)	29 (17)	31 (15)	34 (20)	38 (26)	31 (14)	30 (16)
MCMC	41 (10)	50 (14)	68 (17)	84 (20)	108 (24)	130 (30)	52 (34)	54 (26)	55 (22)
MLP MI	44 (11)	52 (17)	78 (21)	98 (21)	112 (25)	136 (30)	61 (37)	47 (24)	46 (25)
MLP SI	30 (9)	32 (14)	38 (14)	46 (15)	53 (22)	66 (28)	35 (13)	30 (11)	32 (20)
MS	0.04 (0.0001)	0.04 (0.0001)	0.04 (0.0001)	0.04 (0.0001)	0.04 (0.0001)	0.04 (0.0001)	0.04 (0.0001)	0.04 (0.0001)	0.04 (0.0001)
RBF MI	40 (10)	49 (15)	73 (14)	85 (24)	106 (26)	130 (32)	52 (39)	43 (29)	41 (27)
RBF SI	27 (8)	29 (12)	40 (17)	90 (19)	48 (23)	58 (27)	33 (12)	31 (16)	30 (14)
WKNN MI	41 (7)	47 (16)	61 (24)	80 (23)	96 (27)	119 (31)	50 (31)	41 (27)	40 (25)
WKNN SI	28 (8)	29 (12)	33 (15)	32 (14)	37 (21)	42 (26)	36 (18)	35 (14)	35 (22)
ZI	0.01 (0.0002)	0.01 (0.0002)	0.01 (0.0002)	0.01 (0.0002)	0.01 (0.0002)	0.01 (0.0002)	0.01 (0.0002)	0.01 (0.0002)	0.01 (0.0002)

where $P(Y)$ is the probability of the event of interest; $(x_1,x_2,\ldots,x_{100})$ represent different features; $(\beta_0,\beta_1,\beta_2,\beta_3,\beta_4,\beta_5)$ are the model parameters.

All the features in the model were arranged in the random order in all datasets. The differences between the datasets are mainly due to different combinations of feature values and different values of model parameters. We specified different sets of model parameters for different datasets

Step 5: Generate a uniformly distributed random number in the range (0, 1) for each observation. If the random number is greater than the probability of the event of interest, the value of the response variable is 1, otherwise 0.

Appendix G. Simulation of missing data

We deleted values from the complete training data to simulate ignorable and non-ignorable missing observations in a dataset.

MCAR missing values: were generated in following steps.

Step 1: Generate uniformly distributed random number in the interval (0, 1) for each observation.

Step 2: Specify a range of values within the interval (0, 1) depending on the percentage of data to be removed.

Step 3: Remove the observation if the corresponding random number lies within the range.

Non-random (*MAR and MNAR*) *missing values*: For non-random missing data, we have to remove data in such a way so that removed values of variable x_k depends on the variables x_m and x_n.

Step 1: To simulate non-random missing data, we used a model for the non-responsiveness. The model estimates the probability of removal values of a variable x_k. We generate MAR missing data using Eq. (23) and MNAR missing data using Eq. (24)

$$p(x_{ik}) = \frac{1}{1+\exp(-(\beta_0+\beta_m x_{im}+\beta_n x_{in},\ldots))} \quad (23)$$

and

$$p(x_{ik}) = \frac{1}{1+\exp(-(\beta_0+\beta_1 x_{ik}))} \quad (24)$$

where $p(x_{ik})$ is the probability of removal of x_{ik} in the *ith* observation, x_{im} is the value of variable x_m in the *i*th observation, x_{in} is the value of variable x_n in the *i*th observation. $\beta_0,\beta_m,\beta_n,\ldots$ are model parameters.

Step 2: Generate a uniformly distributed random number (R_i) in the interval (0, 1) for each observation of the variable x_k.

Appendix H. More results

See Tables H1–H6.

References

[1] D.B. Rubin, in: Multiple Imputation for Nonresponse in Surveys, Wiley, New York, 1987.

[2] R.J.A. Little, D.B. Rubin, in: Statistical Analysis with Missing Data, Wiley, New York, 1987.

[3] J. Schafer, in: Analysis of Incomplete Multivariate Data, Chapman and Hall, London, 1997.

[4] J. Tuikkala, L.L. Elo, O. Nevalainen, T. Aittokallio, Missing value imputation improves clustering and interpretation of gene expression microarray data, BMC Bioinformatics 9 (2008) 202.

[5] G. Hawthorne, P. Elliott, Imputing cross-sectional missing data: comparison of common techniques, Australian and New Zealand Journal of Psychiatry 37 (7) (2005) 583–590.

[6] G. Carlo, J. Yao, A multiple-imputation metropolis version of the EM algorithm, Biometrika 90 (3) (2003) 643–654.

[7] J.P. Hobert, D. Marchev, A theoretical comparison of the data augmentation, marginal augmentation and PX-DA algorithms, Annals of Statistics 2 (2008) 532–554.

[8] J. Chen, J. Shao, Nearest neighbour imputation for survey data, Journal of Official Statistics 16 (2) (2000) 113–131.

[9] W. Ling, F. Dong-Mei, Estimation of missing values using a weighted *k*-nearest neighbors algorithm, in: Proceedings of the 2009 International Conference on Environmental Science and Information Application Technology, vol. 3, 2009, pp. 660–663.

[10] P.J. Lingras, M. Zhong, S.C. Sharma, Evolutionary regression and neural imputations of missing values, Studies in Fuzziness and Soft Computing 226 (2003) 151–163.
[11] P.J. Zufiria, C. Rivero, EM-based radial basis function training with partial information, Lecture Notes in Computer Science 2415 (2002) 138.
[12] A. Rafat, Locally tuned general regression for learning mixture models using small incomplete data sets with outliers and overlapping classes, Lecture Notes in Computer Science 3177 (2004) 774–779.
[13] D. Tomandi, A. Schober, A modified general regression neural network (MGRNN) with new, efficient training algorithms as a robust 'black-box' tool for data analysis, Neural Networks 14 (8) (2001) 1023–1034.
[14] C.V. Bratu, T. Muresan, R. Potolea, Improving classification performance on real data through imputation, in: Proceedings of 2008 IEEE International Conference on Automation, Quality and Testing, Robotics, Cluj-Napoca, Romania, May 22–25, 2008.
[15] J.K. Chapin, M.A.L. Nicolelis, Principal component analysis of neuronal ensemble activity reveals multidimensional somatosensory representations, Journal of Neuroscience Methods 94 (1) (1999) 121–140.
[16] K. Kim, S. Cho, Ensemble classifiers based on correlation analysis for DNA microarray classification, Neurocomputing 70 (1–3) (2006) 187–199.
[17] S. Dondeti, K. Kannan, R. Manavalan, Genetic algorithm optimized neural networks ensemble for estimation of mefenamic acid and paracetamol in tablets, Acta Chimica Slovenica 52 (2005) 440–449.
[18] Y. Chen, Y. Zhao, A novel ensemble of classifiers for microarray data classification, Applied Soft Computing 8 (4) (2008) 1664–1669.
[19] A. Tsymbal, S. Puuronen, D. Patterson, Feature selection for ensembles of simple Bayesian classifiers, Lecture Notes in Computer Science 2366 (2002) 1611–3349.
[20] E. Bauer, R. Kohavi, An empirical comparison of voting classification algorithms: bagging, boosting, and variants, Machine Learning 36 (1 and 2) (1999) 105–139.
[21] C. Dimitrakakis, S. Bengio, Online adaptive policies for ensemble classifiers, Neurocomputing 64 (2005) 211–221.
[22] Z. Shen, F. Kong, Optimizing weights by genetic algorithm for neural network ensemble, Lecture Notes in Computer Science 3173 (2004) 323–331.
[23] Y. Chen, B. Yang, A. Abraham, Flexible neural trees ensemble for stock index modelling, Neurocomputing 70 (2007) 697–703.
[24] P.K. Hopke, C. Liu, D.B. Rubin, Multiple imputation for multivariate data with missing and below-threshold measurements: time series concentrations of pollutants in the Arctic, Biometrics 57 (1) (2001) 22–33.
[25] D.B. Rubin, in: Multiple Imputations in Sample Surveys, Wiley, New York, 1978.
[26] D.B. Rubin, N. Schenker, Multiple imputation for interval estimation from simple random samples with ignorable response, Journal of the American Statistical Association 81 (394) (1986) 366–374.
[27] J.L. Schafer, M.K. Olsen, Multiple imputation for multivariate missing-data problems: a data analyst's perspective, Multivariate Behavioral Research 33 (4) (1998) 547–571.
[28] R.E. Fay, Alternative paradigms for the analysis of imputed survey data, Journal of the American Statistical Association 91 (434) (1996) 490–498.
[29] J.K. Kim, W. Fuller, Fractional hot deck imputation, Biometrika 91 (3) (2004) 559–578.
[30] D.F. Specht, A general regression neural network, IEEE Transactions on Neural Networks 20 (6) (1991) 568–576.
[31] I.A. Gheyas, L.S. Smith, Feature subset selection in large dimensionality domains, Pattern Recognition 43 (1) (2010) 5–13.
[32] R.M. Valdovinos, J.S. Sanchez, Combining multiple classifiers with dynamic weighted voting, Lecture Notes in Artificial Intelligence 1 (2009) 510–516.
[33] S. Siegel, N.J. Castellan Jr, in: Nonparametric statistics: for the behavioural sciences., 2nd ed., McGraw-Hill, New York, 1988.
[34] U.C.I. Irvine Machine Learning Repository. [Online]. Available: 〈http://archive.ics.uci.edu/ml/〉.

Iffat A. Gheyas recently completed her Ph.D. in Computational Intelligence at the University of Stirling, UK in 2010. Her research focus is on machine learning algorithms for data mining tasks.

Leslie S. Smith received the B.Sc. degree in 1973, and the Ph.D. in 1981, both from Glasgow University. From 1980 to 1983, he was a lecturer at Glasgow University. Since 1984 he has worked at Stirling University, where he is now a Professor of Computing Science and Head of the Department of Computing Science and Mathematics. His research interests are in signal processing for neural systems, engineering approximations to early auditory processing, neural/electronic interfacing and neuromorphic systems. He is a Senior Member of the Institute of Electrical and Electronic Engineers.