Chapter 4
Dealing with Missing Values

Abstract In this chapter the reader is introduced to the approaches used in the literature to tackle the presence of Missing Values (MVs). In real-life data, information is frequently lost in data mining, caused by the presence of missing values in attributes. Several schemes have been studied to overcome the drawbacks produced by missing values in data mining tasks; one of the most well known is based on preprocessing, formally known as imputation. After the introduction in Sect. 4.1, the chapter begins with the theoretical background which analyzes the underlying distribution of the missingness in Sect. 4.2. From this point on, the successive sections go from the simplest approaches in Sect. 4.3, to the most advanced proposals, focusing in the imputation of the MVs. The scope of such advanced methods includes the classic maximum likelihood procedures, like Expectation-Maximization or Multiple-Imputation (Sect. 4.4) and the latest Machine Learning based approaches which use algorithms for classification or regression in order to accomplish the imputation (Sect. 4.5). Finally a comparative experimental study will be carried out in Sect. 4.6.

4.1 Introduction

Many existing, industrial and research data sets contain MVs in their attribute values. Intuitively a MV is just a value for attribute that was not introduced or was lost in the recording process. There are various reasons for their existence, such as manual data entry procedures, equipment errors and incorrect measurements. The presence of such imperfections usually requires a preprocessing stage in which the data is prepared and cleaned [71], in order to be useful to and sufficiently clear for the knowledge extraction process. The simplest way of dealing with MVs is to discard the examples that contain them. However, this method is practical only when the data contains a relatively small number of examples with MVs and when analysis of the complete examples will not lead to serious bias during the inference [54].

MVs make performing data analysis difficult. The presence of MVs can also pose serious problems for researchers. In fact, inappropriate handling of the MVs in the analysis may introduce bias and can result in misleading conclusions being drawn

S. García et al., *Data Preprocessing in Data Mining*,
Intelligent Systems Reference Library 72, DOI 10.1007/978-3-319-10247-4_4

from a research study, and can also limit the generalizability of the research findings [96]. Three types of problems are usually associated with MVs in DM [5]:

1. loss of efficiency;
2. complications in handling and analyzing the data; and
3. bias resulting from differences between missing and complete data.

Recently some authors have tried to estimate how many MVs are needed to noticeably harm the prediction accuracy in classification [45].

Usually the treatment of MVs in DM can be handled in three different ways [27]:

- The first approach is to discard the examples with MVs in their attributes. Therefore deleting attributes with elevated levels of MVs is included in this category too.
- Another approach is the use of maximum likelihood procedures, where the parameters of a model for the complete portion of the data are estimated, and later used for imputation by means of sampling.
- Finally, the imputation of MVs is a class of procedures that aims to fill in the MVs with estimated ones. In most cases, a data set's attributes are not independent from each other. Thus, through the identification of relationships among attributes, MVs can be determined

We will focus our attention on the use of imputation methods. A fundamental advantage of this approach is that the MV treatment is independent of the learning algorithm used. For this reason, the user can select the most appropriate method for each situation faced. There is a broad family of imputation methods, from simple imputation techniques like mean substitution, KNN, etc.; to those which analyze the relationships between attributes such as: SVM-based, clustering-based, logistic regressions, maximum likelihood procedures and multiple imputation [6, 26].

The use of imputation methods for MVs is a task with a well established background. It is possible to track the first formal studies to several decades ago. The work of [54] laid the foundation of further work in this topic, specially in statistics. From their work, imputation techniques based on sampling from estimated data distributions followed, distinguishing between single imputation procedures (like Expectation-Maximization (EM) procedures [81]) and multiple imputation ones [82], the latter being more reliable and powerful but more difficult and restrictive to be applied.

These imputation procedures became very popular for quantitative data, and therefore they were easily adopted in other fields of knowledge, like bioinformatics [49, 62, 93], climatic science [85], medicine [94], etc. The imputation methods proposed in each field are adapted to the common characteristics of the data analyzed in it. With the popularization of the DM field, many studies in the treatment of MVs arose in this topic, particularly in the classification task. Some of the existent imputation procedures of other fields are adapted to be used in classification, for example adapting them to deal with qualitative data, while many specific approaches are proposed.

4.2 Assumptions and Missing Data Mechanisms

It is important to categorize the mechanisms which lead to the introduction of MVs [54]. The assumptions we make about the missingness mechanism and the MVs pattern of MVs can affect which treatment method could be correctly applied, if any.

When thinking about the missing data mechanism the probability distributions that lie beneath the registration of rectangular data sets should be taken into account, where the rows denote different registers, instances or cases, and the columns are the features or variables. A common assumption is that the instances are all independent and identically distributed (i.i.d.) draws of some multivariate probability distribution. This assumption is also made by Schafer in [82] where the schematic representation followed is depicted in Fig. 4.1.

X being the $n \times m$ rectangular matrix of data, we usually denote as x_i the ith row of X. If we consider the i.i.d. assumption, the probability function of the complete data can be written as follows:

$$P(X|\theta) = \prod_{i=1}^{n} f(x_i|\theta), \tag{4.1}$$

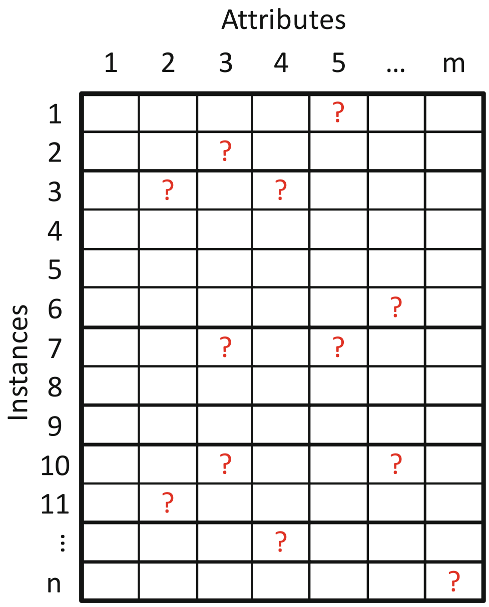

Fig. 4.1 Data set with MVs denoted with a '?'

where f is the probability function for a single case and θ represents the parameters of the model that yield such a particular instance of data. The main problem is that the particular parameters' values θ for the given data are very rarely known. For this reason authors usually overcome this problem by considering distributions that are commonly found in nature and their properties are well known as well. The three distributions that standout among these are:

1. the multivariate normal distribution in the case of only real valued parameters;
2. the multinomial model for cross-classified categorical data (including loglinear models) when the data consists of nominal features; and
3. mixed models for combined normal and categorical features in the data [50, 55].

If we call X_{obs} the observed part of X and we denote the missing part as X_{mis} so that $X = (X_{obs}, X_{mis})$, we can provide a first intuitive definition of what *missing at random* (MAR) means. Informally talking, when the probability that an observation is missing may depend on X_{obs} but not on X_{mis} we can state that the missing data is missing at random.

In the case of MAR missing data mechanism, given a particular value or values for a set of features belonging to X_{obs}, the distribution of the rest of features is the same among the observed cases as it is among the missing cases. Following Schafer's example based on [79], let suppose that we dispose an $n \times p$ matrix called B of variables whose values are 1 or 0 when X elements are observed and missing respectively. The distribution of B should be related to X and to some unknown parameters ζ, so we dispose a probability model for B described by $P(B|X, \zeta)$. Having a MAR assumption means that this distribution does not depend on X_{mis}:

$$P(B|X_{obs}, X_{mis}, \zeta) = P(B|X_{obs}, \zeta). \tag{4.2}$$

Please be aware of MAR does not suggest that the missing data values constitute just another possible sample from the probability distribution. This condition is known as missing completely at random (MCAR). Actually MCAR is a special case of MAR in which the distribution of an example having a MV for an attribute does not depend on either the observed or the unobserved data. Following the previous notation, we can say that

$$P(B|X_{obs}, X_{mis}, \zeta) = P(B|\zeta). \tag{4.3}$$

Although there will generally be some loss of information, comparable results can be obtained with missing data by carrying out the same analysis that would have been performed with no MVs. In practice this means that, under MCAR, the analysis of only those units with complete data gives valid inferences.

Please note that MCAR is more restrictive than MAR. MAR requires only that the MVs behave like a random sample of all values in some particular subclasses defined by observed data. In such a way, MAR allows the probability of a missing datum to depend on the datum itself, but only indirectly through the observed values.

Recently a software package has been published in which the MCAR condition can be tested [43].

A third case arises when MAR does not apply as the MV depends on both the rest of observed values and the proper value itself. That is

$$P(B|X_{obs}, X_{mis}, \zeta) \tag{4.4}$$

is the actual probability estimation. This model is usually called not missing at random (NMAR) or missing not at random (MNAR) in the literature. This model of missingness is a challenge for the user as the only way to obtain an unbiased estimate is to model the missingness as well. This is a very complex task in which we should create a model accounting for the missing data that should be later incorporated to a more complex model used to estimate the MVs. However, even when we cannot account for the missingness model, the introduced bias may be still small enough. In [23] the reader can find an example of how to perform this.

4.3 Simple Approaches to Missing Data

In this section we introduce the most simplistic methods used to deal with MVs. As they are very simple, they usually do not take into account the missingness mechanism and they blindly perform the operation.

The most simple approach is to do not impute (DNI). As its name indicates, all the MVs remain unreplaced, so the DM algorithm must use their default MVs strategies if present. Often the objective is to verify whether imputation methods allow the classification methods to perform better than when using the original data sets. As a guideline, in [37] a previous study of imputation methods is presented. As an alternative for these learning methods that cannot deal with explicit MVs notation (as a special value for instance) another approach is to convert the MVs to a new value (encode them into a new numerical value), but such a simplistic method has been shown to lead to serious inference problems [82].

A very common approach in the specialized literature, even nowadays, is to apply case deletion or ignore missing (IM). Using this method, all instances with at least one MV are discarded from the data set. Although IM often results in a substantial decrease in the sample size available for the analysis, it does have important advantages. In particular, under the assumption that data is MCAR, it leads to unbiased parameter estimates. Unfortunately, even when the data are MCAR there is a loss in power using this approach, especially if we have to rule out a large number of subjects. And when the data is not MCAR, it biases the results. For example when low income individuals are less likely to report their income level, the resulting mean is biased in favor of higher incomes. The alternative approaches discussed below should be considered as a replacement for IM.

Often seen as a good choice, the substitution of the MVs for the global most common attribute value for nominal attributes, and global average value for numerical attributes (MC) [36] is widely used, specially when many instances in the data set contain MVs and to apply DNI would result in a very reduced and unrepresentative pre-processed data set. This method is very simple: for nominal attributes, the MV is replaced with the most common attribute value, and numerical values are replaced with the average of all values of the corresponding attribute.

A variant of MC is the concept most common attribute value for nominal attributes, and concept average value for numerical attributes (CMC) [36]. As stated in *MC*, the MV is replaced by the most repeated one if nominal or is the mean value if numerical, but considers only the instances with the same class as the reference instance.

Older and rarely used DM approaches may be put under this category. For example Hot deck imputation goes back over 50 years and was used quite successfully by the Census Bureau and others. It is referred from time to time [84] and thus it is interesting to describe it here partly for historical reasons and partly because it represents an approach of replacing data that is missing.

Hot deck has it origins in the surveys made in USA in the 40s and 50s, when most people felt impelled to participate in survey filling. As a consequence little data was missing and when any registers were effectively missing, a random complete case from the same survey was used to substitute the MVs. This process can be simulated nowadays by clustering over the complete data, and associating the instance with a cluster. Any complete example from the cluster can be used to fill in the MVs [6]. Cold deck is similar to hot deck, but the cases or instances used to fill in the MVs came from a different source. Traditionally this meant that the case used to fill the data was obtained from a different survey. Some authors have recently assessed the limitations imposed to the donors (the instances used to substitute the MVs) [44].

4.4 Maximum Likelihood Imputation Methods

At the same time Rubin et al. formalized the concept of missing data introduction mechanisms described in Sect. 4.2, they advised against use case deletion as a methodology (IM) to deal with the MVs. However, using MC or CMC techniques are not much better than replacing MVs with fixed values, as they completely ignore the mechanisms that yield the data values. In an ideal and rare case where the parameters of the data distribution θ were known, a sample from such a distribution conditioned to the other attributes' values or not depending of whether the MCAR, MAR or NMAR applies, would be a suitable imputed value for the missing one. The problem is that the parameters θ are rarely known and also very hard to estimate [38].

In a simple case such as flipping a coin, $P(heads) = \theta$ and $P(tails) = 1 - \theta$. Depending on the coin being rigged or not, the value of θ can vary and thus its value is unknown. Our only choice is to flip the coin several times, say n, obtaining h heads and $n - h$ tails. An estimation of θ would be $\widehat{\theta} = h/n$.

More formally, the likelihood of θ is obtained from a binomial distribution $P(\theta) = \binom{h}{n}\theta^h(1-\theta)^{n-h}$. Our $\widehat{\theta}$ can be proven to be the *maximum likelihood* estimate of θ.

So the next question arises: to solve a maximum likelihood type problem, can we analytically maximize the likelihood function? We have shown it can work with one dimensional Bernoulli problems like the coin toss, and that it also works with one dimensional Gaussian by finding the μ and σ parameters. To illustrate the latter case let us assume that we have the samples 1, 4, 7, 9 obtained from a normal distribution and we want to estimate the population mean for the sake of simplicity, that is, in this simplistic case $\theta = \mu$. The maximum likelihood problem here is to choose a specific value of μ and compute $p(1) \cdot p(4) \cdot p(7) \cdot p(9)$. Intuitively one can say that this probability would be very small if we fix $\mu = 10$ and would be higher for $\mu = 4$ or $\mu = 5$. The value of μ that produces the maximum product of combined probabilities is what we call the maximum likelihood estimate of $\mu = \theta$. Again, in our case the maximum likelihood estimate would constitute the sample mean $\mu = 5.25$ and adding the variance to the problem can be solved again using the sample variance as the best estimator.

In real world data things are not that easy. We can have distribution that may not be well behaved or have too many parameters making the actual solution computationally too complex. Having a likelihood function made of a mixture of 100 100-dimensional Gaussians would yield 10,000 parameters and thus direct trial-error maximization is not feasible. The way to deal with such complexity is to introduce hidden variables in order to simplify the likelihood function and, in our case as well, to account for MVs. The observed variables are those that can be directly measured from the data, while hidden variables influence the data but are not trivial to measure. An example of an observed variable would be if it is sunny today, whereas the hidden variable can be $P(sunny\ today|sunny\ yesterday)$.

Even simplifying with hidden variables does not allow us to reach the solution in a single step. The most common approach in these cases would be to use an iterative approach in which we obtain some parameter estimates, we use a regression technique to impute the values and repeat. However as the imputed values will depend on the estimated parameters θ, they will not add any useful information to the process and can be ignored. There are several techniques to obtain maximum likelihood estimators. The most well known and simplistic is the EM algorithm presented in the next section.

4.4.1 Expectation-Maximization (EM)

In a nutshell the EM algorithm estimates the parameters of a probability distribution. In our case this can be achieved from incomplete data. It iteratively maximizes the likelihood of the complete data X_{obs} considered as a function dependent of the parameters [20].

That is, we want to model dependent random variables as the observed variable a and the hidden variable b that generates a. We stated that a set of unknown parameters θ governs the probability distributions $P_\theta(a)$, $P_\theta(b)$. As an iterative process, the EM algorithm consists of two steps that are repeated until convergence: the expectation (E-step) and the maximization (M-step) steps.

The E-step tries to compute the expectation of $logP_\theta(y, x)$:

$$Q(\theta, \theta') = \sum_y P_{\theta'}(b|a) logP_\theta(b, a), \tag{4.5}$$

where θ' are the new distribution parameters. Please note that we are using the *log*. The reason for this is that we need to multiply the probabilities of each observed value for an specific set of parameters. But multiplying several probabilities will soon yield a very small number and thus produce a loss of precision in a computer due to limited digital accuracy. A typical solution is then to use the *log* of these probabilities and to look for the maximum log likelihood. As the logs will be negative, we are looking for the set of parameters whose likelihood is as close to 0 as possible. In the M-step we try to find the θ that maximizes Q.

How can we find the θ that maximizes Q? Let us review conditional expectation where A and B are random variables drawn from distributions $P(a)$ and $P(b)$ to resolve the M-step. The conditional distribution is given by $P(b|a) = \frac{P(b,a)}{P(a)}$ and then $E[B] = \sum_b P(b)b$. For a function depending on B $h(B)$ the expectation is trivially obtained by $E[h(B)] = \sum_b P(b)h(b)$. For a particular value $A(A = a)$ the expectation is $E[h(B)|a] = \sum_b P(b|a)h(b)$.

Remember that we want to pick a θ that maximizes the log likelihood of the observed (a) and the unobserved (b) variables given an observed variable a and the previous parameters θ'. The conditional expectation of $logP_\theta(b, a)$ given a and θ' is

$$E[logP(b, a|\theta)|a, \theta'] = \sum_y P(b|a, \theta') logP(b, a|\theta) \tag{4.6}$$

$$= \sum_y P_{\theta'}(b|a) logP_\theta(b, a). \tag{4.7}$$

The key is that if $\sum_b P_{\theta'}(b|a) logP_\theta(b, a) > \sum_b P_{\theta'}(b|a) logP_{\theta'}(b, a)$ then $P_\theta(a) > P_{\theta'}(a)$. If we can improve the expectation of the log likelihood, EM is improving the model of the observed variable a.

In any real world problem, we do not have a single point but a series of attributes $x_1, \ldots, x_n$. Assuming i.i.d. we can sum over all points to compute the expectation:

$$Q(\theta, \theta') = \sum_{i=1}^{n} \sum_b P_{\theta'}(b|x_i) logP_\theta(b, x_i) \tag{4.8}$$

The EM algorithm is not perfect: it can be stuck in local maxima and also depends on an initial θ value. The latter is usually resolved by using a bootstrap process in order to choose a correct initial θ. Also the reader may have noticed that we have not talked about any imputation yet. The reason is EM is a meta algorithm that it is adapted to a particular application.

To use EM for imputation first we need to choose a plausible set of parameters, that is, we need to assume that the data follows a probability distribution, which is usually seen as a drawback of these kind of methods. The EM algorithm works better with probability distributions that are easy to maximize, as Gaussian mixture models. In [85] an approach of EM using multivariate Gaussian is proposed as using multivariate Gaussian data can be parameterized by the mean and the covariance matrix.

In each iteration of the EM algorithm for imputation the estimates of the mean μ and the covariance Σ are represented by a matrix and revised in three phases. These parameters are used to apply a regression over the MVs by using the complete data. In the first one in each instance with MVs the regression parameters B for the MVs are calculated from the current estimates of the mean and covariance matrix and the available complete data. Next the MVs are imputed with their conditional expectation values from the available complete ones and the estimated regression coefficients

$$x_{mis} = \mu_{mis} + (x_{obs} - \mu_{obs})B + e, \tag{4.9}$$

where the instance x of n attributes is separated into the observed values x_{obs} and the missing ones x_{mis}. The mean and covariance matrix are also separated in such a way. The residual $e \in \mathbb{R}^{1\times n_{mis}}$ is assumed to be a random vector with mean zero and unknown covariance matrix. These two phases would complete the E-step. Please note that for the iteration of the algorithm the imputation is not strictly needed as only the estimates of the mean and covariance matrix are, as well as the regression parameters. But our ultimate goal is to have our data set filled, so we use the latest regression parameters to create the best imputed values so far.

In the third phase the M-step is completed by re-estimating the mean a covariance matrix. The mean is taken as the sample mean of the completed data set and the covariance is the sample covariance matrix and the covariance matrices of the imputation errors as shown in [54]. That is:

$$\widehat{B} = \widehat{\Sigma}^{-1}_{obs,obs}\widehat{\Sigma}_{obs,mis}, \mathit{and} \tag{4.10}$$

$$\widehat{C} = \widehat{\Sigma}_{mis,mis} - \widehat{\Sigma}_{mis,obs}\widehat{\Sigma}^{-1}_{obs,obs}\widehat{\Sigma}_{obs,mis} \tag{4.11}$$

The hat accent $\widehat{A}$ designates an estimate of a quantity A. After updating B and C the mean and covariance matrix must be updated with

$$\widehat{\mu}^{(t+1)} = \frac{1}{n}\sum_{i=1}^{n} X_i \tag{4.12}$$

and

$$\widehat{\Sigma}^{(t+1)} = \frac{1}{\tilde{n}} \sum_{i=1}^{n} \left[\widehat{S}_i^{(t)} - (\widehat{\mu}^{(t+1)})\widehat{\mu}^{(t+1)} \right], \tag{4.13}$$

where, for each instance $x = X_i$, the conditional expectation $\widehat{S}_i^{(t)}$ of the cross-products is composed of three parts. The two parts that involve the available values in the instance,

$$E(x_{obs}^T x_{obs} | x_{obs}; \widehat{\mu}^{(t)}, \widehat{\Sigma}^{(t)}) = x_{obs}^T x_{obs} \tag{4.14}$$

and

$$E(x_{mis}^T x_{mis} | x_{obs}; \widehat{\mu}^{(t)}, \widehat{\Sigma}^{(t)}) = \widehat{x}_{mis}^T \widehat{x}_{mis} + \widehat{C}, \tag{4.15}$$

is the sum of the cross-product of the imputed values and the residual covariance matrix $\widehat{C} = Cov(x_{miss}, x_{mis} | x_{obs}; \widehat{\mu}^{(t)}, \widehat{\Sigma}^{(t)})$, the conditional covariance matrix of the imputation error. The normalization constant $\tilde{n}$ of the covariance matrix estimate [Eq. (4.13)] is the number of degrees of freedom of the sample covariance matrix of the completed data set.

The first estimation of the mean and covariance matrix needs to rely on a completely observed data set. One solution in [85] is to fill the data set with the initial estimates of the mean and covariance matrices. The process ends when the estimates of the mean and covariance matrix do not change over a predefined threshold. Please note that this EM approach is only well suited for numeric data sets, constituting a limitation for the application of EM, although an extension for mixed numerical and nominal attributes can be found in [82].

The EM algorithm is still valid nowadays, but is usually part of a system in which it helps to evolve some distributions like GTM neural networks in [95]. Still some research is being carried out for EM algorithms in which its limitations are being improved and also are applied to new fields like semi-supervised learning [97]. The most well known version of the EM for real valued data sets is the one introduced in [85] where the basic EM algorithm presented is extended with a regularization parameter.

4.4.2 Multiple Imputation

One big problem of the maximum likelihood methods like EM is that they tend to underestimate the inherent errors produced by the estimation process, formally standard errors. The Multiple Imputation (MI) approach was designed to take this into account to be a less biased imputation method, at the cost of being computationally expensive. MI is a Monte Carlo approach described very well by [80] in which we generate multiple imputed values from the observed data in a very similar way to the EM algorithm: it fills the incomplete data by repeatedly solving the observed-data. But a significative difference between the two methods is attained: while EM generates a single imputation in each step from the estimated parameters at each step, MI performs several imputations that yield several complete data sets.

This repeated imputation can be done thanks to the use of Markov Chain Monte Carlo methods, as the several imputations are obtained by introducing a random component, usually from a standard normal distribution. In a more advanced fashion, MI also considers that the parameters estimates are in fact sample estimates. Thus, the parameters are not directly estimated from the available data but, as the process continues, they are drawn from their Bayesian posterior distributions given the data at hand. These assumptions means that only in the case of MCAR or MAR missingness mechanisms hold MI should be applied.

As a result Eq. (4.9) can be applied with slight changes as the e term is now a sample from a standard normal distribution and is applied more than once to obtain several imputed values for a single MV. As indicated in the previous paragraph, MI has a Bayesian nature that forces the user to specify a prior distribution for the parameters θ of the model from which the e term is drawn. In practice [83] is stressed out that the results depend more on the election of the distribution for the data than the distribution for θ. Unlike the single imputation performed by EM where only one imputed value for each MV is created (and thus only one value of e is drawn), MI will create several versions of the data set, where the observed data X_{obs} is essentially the same, but the imputed values for X_{mis} will be different in each data set created. This process is usually known as data augmentation (DA) [91] as depicted in Fig. 4.2.

Surprisingly not many imputation steps are needed. Rubin claims in [80] that only 3–5 steps are usually needed. He states that the efficiency of the final estimation built upon m imputations is approximately:

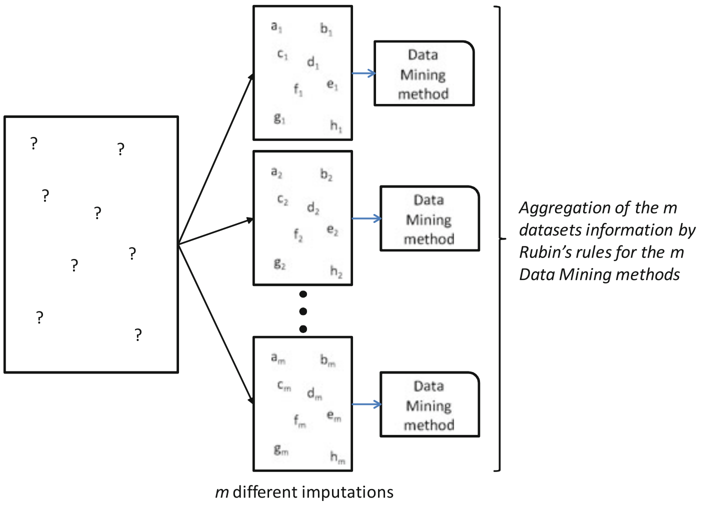
?
a_1 b_1 c_1 d_1 e_1 f_1 g_1 h_1
a_2 b_2 c_2 d_2 e_2 f_2 g_2 h_2
a_m b_m c_m d_m e_m f_m g_m h_m
Data Mining method
Aggregation of the m datasets information by Rubin's rules for the m Data Mining methods
m different imputations

Fig. 4.2 Multiple imputation process by data augmentation. Every MV denoted by a '?' is replaced by several imputed and different values that will be used to continue the process later

$$\left(1+\frac{\gamma}{m}\right)^{-1},$$

where γ is the fraction of missing data in the data set. With a 30 % of MVs in each data set, which is a quite high amount, with 5 different final data sets a 94 % of efficiency will be achieved. Increasing the number to $m = 10$ slightly raises the efficiency to 97 %, which is a low gain paying the double computational effort.

To start we need an estimation of the mean and covariance matrices. A good approach is to take them from a solution provided from an EM algorithm once their values have stabilized at the end of its execution [83]. Then the DA process starts by alternately filling the MVs and then making inferences about the unknown parameters in a stochastic fashion. First DA creates an imputation using the available values of the parameters of the MVs, and then draws new parameter values from the Bayesian posterior distribution using the observed and missing data. Concatenating this process of simulating the MVs and the parameters is what creates a Markov chain that will converge at some point. The distribution of the parameters θ will stabilize to the posterior distribution averaged over the MVs, and the distribution of the MVs will stabilize to a predictive distribution: the proper distribution needed to drawn values for the MIs.

Large rates of MVs in the data sets will cause the convergence to be slow. However, the meaning of *convergence* is different to that used in EM. In EM the parameter estimates have converged when they no longer change from one iteration to the following over a threshold. In DA the distribution of the parameters do no change across iterations but the random parameter values actually continue changing, which makes the convergence of DA more difficult to assess than for EM. In [83] the authors propose to reinterpret convergence in DA in terms of lack of serial dependence: DA can be said to have converged by k cycles if the value of any parameter at iteration $t \in 1, 2, \ldots$ is statistically independent of its value at iteration $t + k$. As the authors show in [83] the DA algorithm usually converges under these terms in equal or less cycles than EM.

The value k is interesting, because it establishes when we should stop performing the Markov chain in order to have MI that are *independent* draws from the missing data predictive distribution. A typical process is to perform m runs, each one of length k. That is, for each imputation from 1 to m we perform the DA process during k cycles. It is a good idea not to be too conservative with the k value, as after convergence the process remains stationary, whereas with low k values the m imputed data sets will not be truly independent. Remember that we do not need a high m value, so k acts as the true computational effort measure.

Once the m MI data sets have been created, they can be analyzed by any standard complete-data methods. For example, we can use a linear or logistic regression, a classifier or any other technique applied to each one of the m data sets, and the variability of the m results obtained will reflect the uncertainty of MVs. It is common to combine the results following the rules provided by Rubin [80] that act as measures of ordinary sample variation to obtain a single inferential statement of the parameters of interest.

Rubin's rules to obtain an overall set of estimated coefficients and standard errors proceed as follows. Let $\widehat{R}$ denote the estimation of interest and U its estimated variance, R being either an estimated regression coefficient or a kernel parameter of a SVM, whatever applies. Once the MIs have been obtained, we will have $\widehat{R}_1, \widehat{R}_2, \ldots, \widehat{R}_m$ estimates and their respective variances $U_1, U_2, \ldots, U_m$. The overall estimate, occasionally called the MI estimate is given by

$$\overline{R} = \frac{1}{m}\sum_{i=1}^{m}\widehat{R}_i. \tag{4.16}$$

The variance for the estimate has two components: the variability within each data set and across data sets. The within imputation variance is simply the average of the estimated variances:

$$\overline{U} = \frac{1}{m}\sum_{i=1}^{m}U_i, \tag{4.17}$$

whereas the between imputation variance is the sample variance of the proper estimates:

$$B = \frac{1}{m-1}\sum_{i=1}^{m}(\widehat{R}_i - \overline{R})^2. \tag{4.18}$$

The total variance T is the corrected sum of these two components with a factor that accounts for the simulation error in $\widehat{R}$,

$$T = \widehat{U} + \left(1 + \frac{1}{m}\right)B. \tag{4.19}$$

The square root of T is the overall standard error associated to $\overline{R}$. In the case of no MVs being present in the original data set, all $\widehat{R}_1, \widehat{R}_2, \ldots, \widehat{R}_m$ would be the same, then $B = 0$ and $T = \overline{U}$. The magnitude of B with respect to $\overline{U}$ indicates how much information is contained in the missing portion of the data set relative to the observed part.

In [83] the authors elaborate more on the confidence intervals extracted from $\overline{R}$ and how to test the null hypothesis of $R = 0$ by comparing the ratio $\frac{\overline{R}}{\sqrt{T}}$ with a Student's t-distribution with degrees of freedom

$$df = (m-1)\left(1 + \frac{m\overline{U}}{(m+1)B}\right)^2, \tag{4.20}$$

in the case the readers would like to further their knowledge on how to use this hypothesis to check whether the number of MI m was large enough.

The MI algorithm has been widely used in many research fields. Focusing on DM methods to increase the robustness of MI [19], alleviate the parameter selection process [35] and improve Rubin's rules to aggregate models have been proposed [86]. New extensions to new problems like one-class [48] can be found, as well as hybridizations with innovative techniques such as Gray System Theory [92]. Implementing MI is not trivial and reputed implementations can be found in statistical packages as R [9] (see Chap. 10) and Stata [78].

4.4.3 Bayesian Principal Component Analysis (BPCA)

The MV estimation method based on BPCA [62] consists of three elementary processes. They are (1) principal component (PC) regression, (2) Bayesian estimation, and (3) an EM-like repetitive algorithm. In the following we describe each of these processes.

4.4.3.1 PC Regression

For the time being, we consider a situation where there is no MV. PCA represents the variation of D-dimensional example vectors y as a linear combination of principal axis vectors $w_l (1 \leq l \leq K)$ whose number is relatively small ($K < D$):

$$y = \sum_{l=1}^{K} x_l w_l + \epsilon \tag{4.21}$$

The linear coefficients $x_l (1 \leq l \leq K)$ are called factor scores. ϵ denotes the residual error. Using a specifically determined number K, PCA obtains x_l and w_l such that the sum of squared error $\parallel \epsilon \parallel^2$ over the whole data set Y is minimized.

When there is no MV, x_l and w_l are calculated as follows. A covariance matrix S for the example vectors $y_i (1 \leq i \leq N)$ is given by

$$S = \frac{1}{N} \sum_{i=1}^{N} (y_i - \mu)(y_i - \mu)^T, \tag{4.22}$$

where μ is the mean vector of y: $\mu = (1/N) \sum_{i=1}^{N} y_i$. T denotes the transpose of a vector or a matrix. For description convenience, Y is assumed to be row-wisely normalized by a preprocess, so that $\mu = 0$ holds. With this normalization, the result by PCA is identical to that by SVD.

Let $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_D$ and $u_1, u_2, \ldots, u_D$ denote the eigenvalues and the corresponding eigenvectors, respectively, of S. We also define the lth principal axis vector by $w_l = \sqrt{\lambda_l u_l}$ .With these notations, the lth factor score for an example vector

y is given by $x_l = (w_l/\lambda_l)^T y$. Now we assume the existence of MVs. In PC regression, the missing part y_{miss} in the expression vector y is estimated from the observed part y_{obs} by using the PCA result. Let w^l_{obs} and w^l_{miss} be parts of each principal axis w_l, corresponding to the observed and missing parts, respectively, in y. Similarly, let $W = (W_{obs}, W_{miss})$ where W_{obs} or W_{miss} denotes a matrix whose column vectors are $w^1_{obs}, \dots, w^K_{obs}$ or $w^1_{miss}, \dots, w^K_{miss}$, respectively.

Factor scores $x = (x_1, \dots, x_K)$ for the example vector y are obtained by minimization of the residual error

$$err = \parallel y_{obs} - W_{obs}x \parallel^2 .$$

This is a well-known regression problem, and the least square solution is given by

$$x = (W^{obsT} W_{obs})^{-1} W^{obsT} y_{obs}.$$

Using x, the missing part is estimated as

$$y_{miss} = W_{miss}x \tag{4.23}$$

In the PC regression above, W should be known beforehand. Later, we will discuss the way to determine the parameter.

4.4.3.2 Bayesian Estimation

A parametric probabilistic model, which is called probabilistic PCA (PPCA), has been proposed recently. The probabilistic model is based on the assumption that the residual error ϵ and the factor scores $x_l (1 \leq l \leq K)$ in Equation (reflinearcomb) obey normal distributions:

$$\begin{aligned} p(x) &= \mathcal{N}_K(x|0, I_K), \\ p(\epsilon) &= \mathcal{N}_D(\epsilon|0, (1/\tau)I_D), \end{aligned}$$

where $\mathcal{N}_K(x|\mu, \Sigma)$ denotes a K-dimensional normal distribution for x, whose mean and covariance are μ and Σ, respectively. I_K is a $(K \times K)$ identity matrix and τ is a scalar inverse variance of ϵ. In this PPCA model, a complete log-likelihood function is written as:

$$\begin{aligned} ln\, p(y, x|\theta) &\equiv ln\, p(y, x|W, \mu, \tau) \\ &= -\frac{\tau}{2} \parallel y - Wx - \tau \parallel^2 - \frac{1}{2} \parallel x \parallel^2 + \frac{D}{2} ln\, \tau - \frac{K+D}{2} ln 2\Pi, \end{aligned}$$

where $\theta \equiv W, \mu, \tau$ is the parameter set. Since the maximum likelihood estimation of the PPCA is identical to PCA, PPCA is a natural extension of PCA to a probabilistic model.

We present here a Bayesian estimation method for PPCA from the authors. Bayesian estimation obtains the posterior distribution of θ and X, according to the Bayes' theorem:

$$p(\theta, X|Y) \propto p(Y, X|\theta)p(\theta). \tag{4.24}$$

$p(\theta)$ is called a prior distribution, which denotes a priori preference for parameter θ. The prior distribution is a part of the model and must be defined before estimation. We assume conjugate priors for τ and μ, and a hierarchical prior for W, namely, the prior for W, $p(W|\tau, \alpha)$, is parameterized by a hyperparameter $\alpha \in \mathbb{R}^K$.

$$\begin{aligned} p(\theta|\alpha) \equiv p(\mu, W, \tau|\alpha) &= p(\mu|\tau)p(\tau)\prod_{j=1}^{K} p(w_j|\tau, \alpha_j), \\ p(\mu|tau) &= \mathcal{N}(\mu|\overline{\mu}_0, (\gamma_{\mu_0}^{\tau})^{-1} I_m), \\ p(w_j|\tau, \alpha_j) &= \mathcal{N}(w_j|0, (\alpha_j\tau)^{-1} I_m), \\ p(\tau) &= \mathcal{G}(\tau|\overline{\tau}_0, \gamma_{\tau_0}) \end{aligned}$$

$\mathcal{G}(\tau|\overline{\tau}, \gamma_\tau)$ denotes a Gamma distribution with hyperparameters $\overline{\tau}$ and γ_τ:

$$\mathcal{G}(\tau|\overline{\tau}, \gamma_\tau) \equiv \frac{(\gamma_\tau \overline{\tau}^{-1})^{\gamma_\tau}}{\Gamma(\gamma_\tau)} \exp\left[-\gamma_\tau \overline{\tau}^{-1}\tau + (\gamma_\tau - 1)ln\tau\right]$$

where $\Gamma(\cdot)$ is a Gamma function.

The variables used in the above priors, $\gamma_{\mu 0}$, $\overline{\mu}_0$, $\gamma_{\tau 0}$ and $\overline{\tau}_0$ are deterministic hyperparameters that define the prior. Their actual values should be given before the estimation. We set $\gamma_{\mu 0} = \gamma_{\tau 0} = 10^{-10}$, $\overline{\mu}_0 = 0$ and $\overline{\tau}_0 = 1$, which corresponds to an almost non-informative prior.

Assuming the priors and given a whole data set $Y = y$, the type-II maximum likelihood hyperparameter α_{ML-II} and the posterior distribution of the parameter, $q(\theta) = p(\theta|Y, \alpha_{ML-II})$, are obtained by Bayesian estimation.

The hierarchical prior $p(W|\alpha, \tau)$, which is called an automatic relevance determination (ARD) prior, has an important role in BPCA. The jth principal axis w_j has a Gaussian prior, and its variance $1/(\alpha_j\tau)$ is controlled by a hyperparameter α_j which is determined by type-II maximum likelihood estimation from the data. When the Euclidian norm of the principal axis, $\| w_j \|$, is small relatively to the noise variance $1/\tau$, the hyperparameter α_j gets large and the principal axis w_j shrinks nearly to be 0. Thus, redundant principal axes are automatically suppressed.

4.4.3.3 EM-Like Repetitive Algorithm

If we know the true parameter θ_{true}, the posterior of the MVs is given by

$$q(Y_{miss}) = p(Y_{miss}|Y_{obs}, \theta_{true}),$$

which produces equivalent estimation to the PC regression. Here, $p(Y_{miss}|Y_{obs}, \theta_{true})$ is obtained by marginalizing the likelihood (4.24) with respect to the observed variables Y_{obs}. If we have the parameter posterior $q(\theta)$ instead of the true parameter, the posterior of the MVs is given by

$$q(Y_{miss}) = \int d\theta q(\theta) p(Y_{miss}|Y_{obs}, \theta),$$

which corresponds to the Bayesian PC regression. Since we do not know the true parameter naturally, we conduct the BPCA. Although the parameter posterior $q(\theta)$ can be easily obtained by the Bayesian estimation when a complete data set Y is available, we assume that only a part of Y, Y_{obs}, is observed and the rest Y_{miss} is missing. In that situation, it is required to obtain $q(\theta)$ and $q(Y_{miss})$ simultaneously.

We use a variational Bayes (VB) algorithm, in order to execute Bayesian estimation for both model parameter θ and MVs Y_{miss}. Although the VB algorithm resembles the EM algorithm that obtains maximum likelihood estimators for θ and Y_{miss}, it obtains the posterior distributions for θ and Y_{miss}, $q(\theta)$ and $q(Y_{miss})$, by a repetitive algorithm.

The VB algorithm is implemented as follows: (a) the posterior distribution of MVs, $q(Y_{miss})$, is initialized by imputing each of the MVs to instance-wise average; (b) the posterior distribution of the parameter θ, $q(\theta)$, is estimated using the observed data Y_{obs} and the current posterior distribution of MVs, $q(Y_{miss})$; (c) the posterior distribution of the MVs, $q(Y_{miss})$, is estimated using the current $q(\theta)$; (d) the hyperparameter α is updated using both of the current $q(\theta)$ and the current $q(Y_{miss})$; (e) repeat (b)–(d) until convergence.

The VB algorithm has been proved to converge to a locally optimal solution. Although the convergence to the global optimum is not guaranteed, the VB algorithm for BPCA almost always converges to a single solution. This is probably because the objective function of BPCA has a simple landscape. As a consequence of the VB algorithm, therefore, $q(\theta)$ and $q(Y_{miss})$ are expected to approach the global optimal posteriors.

Then, the MVs in the expression matrix are imputed to the expectation with respect to the estimated posterior distribution:

$$\widehat{Y}_{miss} = \int y_{miss} q(Y_{miss}) dY_{miss}. \tag{4.25}$$

4.5 Imputation of Missing Values. Machine Learning Based Methods

The imputation methods presented in Sect. 4.4 originated from statistics application and thus they model the relationship between the values by searching for the hidden distribution probabilities. In Artificial Intelligence modeling the unknown relationships between attributes and the inference of the implicit information contained in a sample data set has been done using ML models. Immediately many authors noticed that the same process that can be carried out to predict a continuous or a nominal value from a previous learning process in regression or classification can be applied to predict the MVs. The use of ML methods for imputation alleviates us from searching for the estimated underlying distribution of the data, but they are still subject to the MAR assumption in order to correctly apply them.

Batista [6] tested the classification accuracy of two popular classifiers (C4.5 and CN2) considering the proposal of KNN as an imputation (KNNI) method and MC. Both CN2 and C4.5 (like [37]) algorithms have their own MV estimation. From their study, KNNI results in good accuracy, but only when the attributes are not highly correlated to each other. Related to this work, [1] have investigated the effect of four methods that deal with MVs. As in [6], they use KNNI and two other imputation methods (MC and median imputation). They also use the KNN and Linear Discriminant Analysis classifiers. The results of their study show that no significantly harmful effect in accuracy is obtained from the imputation procedure. In addition to this, they state that the KNNI method is more robust with the increment of MVs in the data set in respect to the other compared methods.

The idea of using ML or Soft Computing techniques as imputation methods spread from this point on. Li et al. [53] uses a fuzzy clustering method: the Fuzzy K-Means (FKMI). They compare the FKMI with Mean substitution and KMI (K-Means imputation). Using a Root Mean Square Error error analysis, they state that the basic KMI algorithm outperforms the MC method. Experiments also show that the overall performance of the FKMI method is better than the basic KMI method, particularly when the percentage of MVs is high. Feng et al. [29] uses an SVM for filling in MVs (SVMI) but they do not compare this with any other imputation methods. Furthermore, they state that we should select enough complete examples without MVs as the training data set in this case.

In the following we proceed to describe the main details of the most used imputation methods based on ML techniques. We have tried to stay as close as possible to the original notation used by the authors so the interested reader can easily continue his or her exploration of details in the corresponding paper.

4.5.1 Imputation with K-Nearest Neighbor (KNNI)

Using this instance-based algorithm, every time an MV is found in a current instance, KNNI computes the KNN and a value from them is imputed. For nominal values, the most common value among all neighbors is taken, and for numerical values the average value is used. Therefore, a proximity measure between instances is needed for it to be defined. The Euclidean distance (it is a case of a L_p norm distance) is the most commonly used in the literature.

In order to estimate a MV y_{ih} in the ith example vector y_i by KNNI [6], we first select K examples whose attribute values are similar to y_i. Next, the MV is estimated as the average of the corresponding entries in the selected K expression vectors. When there are other MVs in y_i and/or y_j, their treatment requires some heuristics. The missing entry y_{ih} is estimated as average:

$$y_{\hat{i}h} = \frac{\sum_{j \in I_{Kih}} y_{jh}}{|I_{Kih}|}, \tag{4.26}$$

where I_{Kih} is now the index set of KNN examples of the ith example, and if y_{jh} is missing the jth attribute is excluded from I_{Kih}. Note that KNNI has no theoretical criteria for selecting the best K-value and the K-value has to be determined empirically.

4.5.2 *Weighted Imputation with K-Nearest Neighbour (WKNNI)*

The Weighted KNN method [93] selects the instances with similar values (in terms of distance) to incomplete instance, so it can impute as *KNNI* does. However, the estimated value now takes into account the different distances to the neighbors, using a weighted mean or the most repeated value according to a similarity measure. The similarity measure $s_i(y_j)$ between two examples y_i and y_j is defined by the Euclidian distance calculated over observed attributes in y_i. Next we define the measure as follows:

$$1/s_i = \sum_{h_i \in O_i \bigcap O_j} (y_{ih} - y_{jh})^2, \tag{4.27}$$

where $O_i = \{h|$ the hth component of$y_i$is observed$\}$.

The missing entry y_{ih} is estimated as average weighted by the similarity measure:

$$y_{\hat{i}h} = \frac{\sum_{j \in I_{Kih}} s_i(y_j) y_{jh}}{\sum_{j \in I_{Kih}} s_i(y_j)}, \tag{4.28}$$

where I_{Kih} is the index set of KNN examples of the ith example, and if y_{jh} is missing the jth attribute is excluded from I_{Kih}. Note that KNNI has no theoretical criteria for selecting the best K-value and the K-value has to be determined empirically.

4.5.3 K-means Clustering Imputation (KMI)

In K-means clustering [53], the intra-cluster dissimilarity is measured by the summation of distances between the objects and the centroid of the cluster they are assigned to. A cluster centroid represents the mean value of the objects in the cluster. Given a set of objects, the overall objective of clustering is to divide the data set into groups based on the similarity of objects, and to minimize the intra-cluster dissimilarity. KMI measures the intra-cluster dissimilarity by the addition of distances among the objects and the centroid of the cluster which they are assigned to. A cluster centroid represents the mean value of the objects in the cluster. Once the clusters have converged, the last process is to fill in all the non-reference attributes for each incomplete object based on the cluster information. Data objects that belong to the same cluster are taken to be nearest neighbors of each other, and KMI applies a nearest neighbor algorithm to replace MVs, in a similar way to KNNI.

Given a set of N objects $X = x_1, x_2, ldots, x_N$ where each object has S attributes, we use $x_{ij}(1 \leq i \leq N \text{and} 1 \leq j \leq S)$ to denote the value of attribute j in object x_i. Object x_i is called a *complete* object, if $\{x_{ij} \neq \phi | \forall 1 \leq j \leq S\}$, and an incomplete object, if $\{x_{ij} = \phi | \exists 1 \leq j \leq S\}$, and we say object x_i has a MV on attribute j. For any incomplete object x_i, we use $R = \{j | x_{ij} \neq \phi, 1 \leq j \leq S\}$ to denote the set of attributes whose values are available, and these attributes are called *reference* attributes. Our objective is to obtain the values of non-reference attributes for the incomplete objects. By K-means clustering method, we divide data set X into K clusters, and each cluster is represented by the centroid of the set of objects in the cluster. Let $V = v_1, \ldots, v_k$ be the set of K clusters, where $v_k(1 \leq k \leq K)$ represents the centroid of cluster k. Note that v_k is also a vector in a S-dimensional space. We use $d(v_k, x_i)$ to denote the distance between centroid v_k and object x_i.

KMI can be divided into three processes. First, randomly select K complete data objects as K centroids. Second, iteratively modify the partition to reduce the sum of the distances for each object from the centroid of the cluster to which the object belongs. The process terminates once the summation of distances is less than a user-specified threshold $\varepsilon = 100$, or no change on the centroids were made in last iteration. The last process is to fill in all the non-reference attributes for each incomplete object based on the cluster information. Data objects that belong to the same cluster are taken as nearest neighbors of each other, and we apply a nearest neighbor algorithm to replace missing data. We use as a distance measure the Euclidean distance.

4.5.4 Imputation with Fuzzy K-means Clustering (FKMI)

In fuzzy clustering, each data object has a membership function which describes the degree to which this data object belongs to a certain cluster. Now we want to extend the original K-means clustering method to a fuzzy version to impute missing data [1, 53]. The reason for applying the fuzzy approach is that fuzzy clustering provides a better description tool when the clusters are not well-separated, as is the case in missing data imputation. Moreover, the original K-means clustering may be trapped in a local minimum status if the initial points are not selected properly. However, continuous membership values in fuzzy clustering make the resulting algorithms less susceptible to get stuck in a local minimum situation.

In fuzzy clustering, each data object x_i has a membership function which describes the degree to which this data object belongs to certain cluster v_k. The membership function is defined in the next equation

$$U(v_k, x_i) = \frac{d(v_k, x_i)^{-27(m-1)}}{\sum_{j=1}^{K} d(v_j, x_i)^{-2/(m-1)}} \tag{4.29}$$

where $m > 1$ is the fuzzifier, and $\sum_{j=1}^{K} U(v_j, x_i) = 1$ for any data object $x_i (1 \leq i \leq N)$. Now we can not simply compute the cluster centroids by the mean values. Instead, we need to consider the membership degree of each data object. Equation (4.30) provides the formula for cluster centroid computation:

$$v_k = \frac{\sum_{i=1}^{N} U(v_k, x_i) \times x_i}{\sum_{i=1}^{N} U(v_k, x_i)} \tag{4.30}$$

Since there are unavailable data in incomplete objects, we use only reference attributes to compute the cluster centroids.

The algorithm for missing data imputation with fuzzy K-means clustering method also has three processes. Note that in the initialization process, we pick K centroids which are evenly distributed to avoid local minimum situation. In the second process, we iteratively update membership functions and centroids until the overall distance meets the user-specified distance threshold ε. In this process, we cannot assign the data object to a concrete cluster represented by a cluster centroid (as did in the basic K-mean clustering algorithm), because each data object belongs to all K clusters with different membership degrees. Finally, we impute non-reference attributes for each incomplete object. We replace non-reference attributes for each incomplete data object x_i based on the information about membership degrees and the values of cluster centroids, as shown in next equation:

$$x_{i,j} = \sum_{k=1}^{K} U(x_i, v_k) \times v_{k,j}, \text{ for any non-reference attribute } j \notin R \tag{4.31}$$

4.5.5 Support Vector Machines Imputation (SVMI)

Support Vector Machines Imputation [29] is an SVM regression based algorithm to fill in MVs, i.e. set the decision attributes (output or classes) as the condition attributes (input attributes) and the condition attributes as the decision attributes, so SVM regression can be used to predict the missing condition attribute values. SVM regression estimation seeks to estimate functions

$$f(x) = (wx) + b, \qquad w, x \in \mathbb{R}^n, b \in \mathbb{R} \tag{4.32}$$

based on data

$$(x_1, y_1), \dots, (x_l, y_l) \in \mathbb{R} \times \mathbb{R} \tag{4.33}$$

by minimizing the regularized risk functional

$$\parallel W \parallel^2 /2 + C \bullet R^{\varepsilon}_{emp} \tag{4.34}$$

where C is a constant determining the trade-off between minimizing the training error, or empirical risk

$$R^{\varepsilon}_{emp} = \frac{1}{l} \sum_{i=1}^{l} |y_i - f(x_i)|_{\varepsilon} \tag{4.35}$$

and the model complexity term $\parallel W \parallel^2$. Here, we use the so-called ε-insensitive loss function

$$|y - f(x)|_{\varepsilon} = \max\{0, |y - f(x)| - \varepsilon\} \tag{4.36}$$

The main insight of the statistical learning theory is that in order to obtain a small risk, one needs to control both training error and model complexity, i.e. explain the data with a simple model. The minimization of Eq. (4.36) is equivalent to the following constrained optimization problem [17]: minimize

$$\tau(w, \xi^{(*)}) = \frac{1}{2} \parallel w \parallel^2 + C \frac{1}{l} \sum_{i=1}^{l} (\xi_i + \xi_i^*) \tag{4.37}$$

subject to the following constraints

$$((w \bullet x_i) + b) - y_i \le \varepsilon + \xi_i \tag{4.38}$$

$$y_i - ((w \bullet x_i) + b) \le \varepsilon + \xi_i^* \tag{4.39}$$

$$\xi_i^{(*)} \ge 0, \quad \varepsilon \ge 0 \tag{4.40}$$

As mentioned above, at each point x_i we allow an error of magnitude ε. Errors above ε are captured by the slack variables ξ^* (see constraints 4.38 and 4.39). They are penalized in the objective function via the regularization parameter C chosen a priori.

In the ν-SVM the size of ε is not defined a priori but is itself a variable. Its value is traded off against model complexity and slack variables via a constant $\nu \in (0, 1]$ minimize

$$\tau(W, \xi^{(*)}, \varepsilon) = \frac{1}{2} \parallel W \parallel^2 + C \bullet (\nu\varepsilon + \frac{1}{l}\sum_{i=1}^{l}(\xi_i + \xi_i^*)) \tag{4.41}$$

subject to the constraints 4.38–4.40. Using Lagrange multipliers techniques, one can show [17] that the minimization of Eq. (4.37) under the constraints 4.38–4.40 results in a convex optimization problem with a global minimum. The same is true for the optimization problem 4.41 under the constraints 4.38–4.40. At the optimum, the regression estimate can be shown to take the form

$$f(x) = \sum_{i=1}^{l}(\alpha_i^* - \alpha_i)(x_i \bullet x) + b \tag{4.42}$$

In most cases, only a subset of the coefficients $(\alpha_i^* - \alpha_i)$ will be nonzero. The corresponding examples x_i are termed support vectors (SVs). The coefficients and the SVs, as well as the offset b; are computed by the ν-SVM algorithm. In order to move from linear (as in Eq. 4.42) to nonlinear functions the following generalization can be done: we map the input vectors x_i into a high-dimensional feature space Z through some chosen a priori nonlinear mapping $\Phi : X_i \rightarrow Z_i$. We then solve the optimization problem 4.41 in the feature space Z. In this case, the inner product of the input vectors $(x_i \bullet x)$ in Eq. (4.42) is replaced by the inner product of their icons in feature space Z, $(\Phi(x_i) \bullet \Phi(x))$. The calculation of the inner product in a high-dimensional space is computationally very expensive. Nevertheless, under general conditions (see [17] and references therein) these expensive calculations can be reduced significantly by using a suitable function k such that

$$(\Phi(x_i) \bullet \Phi(x)) = k(x_i \bullet x), \tag{4.43}$$

leading to nonlinear regressions functions of the form:

$$f(x) = \sum_{i=1}^{l}(\alpha_i^* - \alpha_i)k(x_i, x) + b \tag{4.44}$$

The nonlinear function k is called a kernel [17]. We mostly use a Gaussian kernel

$$k(x, y) \vDash \exp(- \parallel x - y \parallel^2 /(2\sigma_{kernel}^2)) \tag{4.45}$$

We can use SVM regression [29] to predict the missing condition attribute values. In order to do that, first we select the examples in which there are no missing attribute values. In the next step we set one of the condition attributes (input attribute), some of those values are missing, as the decision attribute (output attribute), and the decision attributes as the condition attributes by contraries. Finally, we use SVM regression to predict the decision attribute values.

4.5.6 Event Covering (EC)

Based on the work of Wong et al. [99], a mixed-mode probability model is approximated by a discrete one. First, we discretize the continuous components using a minimum loss of information criterion. Treating a mixed-mode feature n-tuple as a discrete-valued one, the authors propose a new statistical approach for synthesis of knowledge based on cluster analysis: (1) detect from data patterns which indicate statistical interdependency; (2) group the given data into clusters based on detected interdependency; and (3) interpret the underlying patterns for each of the clusters identified. The method of synthesis is based on author's *event–covering* approach. With the developed inference method, we are able to estimate the MVs in the data.

The cluster initiation process involves the analysis of the nearest neighbour distance distribution on a subset of samples, the selection of which is based on a mean probability criterion. Let $X = (X_1, X_2, \dots, X_n)$ be a random n-tuple of related variables and $x = (x_1, x_2, \dots, x_n)$ be its realization. Then a sample can be represented as x. Let S be an ensemble of observed samples represented as n-tuples. The nearest-neighbour distance of a sample x_i to a set of examples S is defined as:

$$D(x_i, S) = min_{x_j \in S_{x_i \neq x_j}} d(x_i, x_j) \tag{4.46}$$

where $d(x_i, x_j)$ is a distance measure. Since we are using discrete values, we have adopted the Hamming distance. Let C be a set of examples forming a simple cluster. We define the maximum within-cluster nearest-neighbour distance as

$$D_c^* = max_{x_i \in C} D(x_i, C) \tag{4.47}$$

D_c^* reflects an interesting characteristic of the cluster configuration: that is, the smaller the D_c^*, the denser the cluster.

Using a mean probability criterion to select a similar subset of examples, the isolated samples can be easily detected by observing the wide gaps in the nearest-neighbour distance space. The probability distribution from which the criterion is derived for the samples can be estimated using a second-order probability estimation. An estimation of $P(x)$ known as the *dependence tree product approximation* can be expressed as:

$$\widehat{P}(x) = \prod_{j=1}^{n} P(x_{mj}|x_{m_{k(j)}}), 0 < k(j) < 1 \tag{4.48}$$

where (1) the index set $m_1, m_2, \ldots, m_n$ is a permutation of the integer set $1, 2, \ldots, n$, (2) the ordered pairs $x_{mj}, x_{m_{k(j)}}$ are chosen so that they the set of branches of a spanning tree defined on X with their summed MI maximized, and (3) $P(x_{m1}|x_{m0}) = P(x_{m1})$. The probability defined above is known to be the best second-order approximation of the high-order probability distribution. Then corresponding to each x in the ensemble, a probability $P(x)$ can be estimated.

In general, it is more likely for samples of relatively high probability to form clusters. By introducing the mean probability below, samples can be divided into two subsets: those above the mean and those below. Samples above the mean will be considered first for cluster initiation.

Let $S = x$. The mean probability is defined as

$$\mu_s = \sum_{x \in S} P(x)/|S| \tag{4.49}$$

where $|S|$ is the number of samples in S. For more details in the probability estimation with *dependence tree product approximation*, please refer to [13].

When distance is considered for cluster initiation, we can use the following criteria in assigning a sample x to a cluster.

1. If there exists more than one cluster, say $C_k|k = 1, 2, \ldots$, such that $D(x, C_k) \leq D^*$ for all k, then all these clusters can be merged together.
2. If exactly one cluster C_k exists, such that $D(x, C_k) \leq D^*$, then x can be grouped into C_k.
3. If $D(x, C_K) > D^*$ for all clusters C_k, then x may not belong to any cluster.

To avoid including distance calculation of outliers, we use a simple method suggested in [99] which assigns D^* the maximum value of all nearest-neighbor distances in L provided there is a sample in L having a nearest-neighbor distance value of $D^* - 1$ (with the distance values rounded to the nearest integer value).

After finding the initial clusters along with their membership, the regrouping process is thus essentially an inference process for estimating the cluster label of a sample. Let $C = a_{c1}, a_{c2}, \ldots, a_{cq}$ be the set of labels for all possible clusters to which x can be assigned. For X_k in X, we can form a contingency table between X_k and C. Let a_{ks} and a_{cj} be possible outcomes of X_k and C respectively, and let $obs(a_{ks}$ and $obsa_{cj}$ be the respectively marginal frequencies of their observed occurrences. The expected relative frequency of (a_{ks}, a_{cj}) is expressed as:

$$exp(a_{ks}, a_{cj}) = \frac{obs(a_{ks}) \times obs(a_{cj})}{|S|} \tag{4.50}$$

Let $obs(a_{ks}, a_{cj})$ represent the actual observed frequency of (a_{ks}, a_{cj}) in S. The expression

$$D = \sum_{j=1}^{q} \frac{(obs_{ks} - exp(a_{ks}, a_{cj}))^2}{exp(a_{ks}, a_{cj})} \tag{4.51}$$

summing over the outcomes of C in the contingency table, possesses an asymptotic chi-squared property with $(q-1)$ degrees of freedom. D can then be used in a criterion for testing the statistical dependency between a_{ks}, and C at a presumed significant level as described below. For this purpose, we define a mapping

$$h_k^c(a_{ks}, C) = \begin{cases} 1, & \text{if } D > \chi^2(q-1); \\ 0, & \text{otherwise.} \end{cases} \tag{4.52}$$

where $\chi^2(q-1)$ is the tabulated chi-squared value. The subset of selected events of X_k, which has statistical interdependency with C, is defined as

$$E_k^c = \left\{ a_{ks} | h_k^c(a_{ks}, C) = 1 \right\} \tag{4.53}$$

We call E_k^c the covered event subset of X_k with respect to C. Likewise, the covered event subset E_c^k of C with respect to X_k can be defined. After finding the covered event subsets of E_c^k and E_k^c between a variable pair (C, X_k), information measures can be used to detect the statistical pattern of these subsets. An interdependence redundancy measure between X_k^c and C^k can be defined as

$$R(X_k^c, C^k) = \frac{I(X_k^c, C^k)}{H(X_k^c, C^k)} \tag{4.54}$$

where $I(X_k^c, C^k)$ is the expected MI and $H(X_k^c, C^k)$ is the Shannon's entropy defined respectively on X_k^c and C^k:

$$I(X_k^c, C^k) = \sum_{a_{cu} \in E_c^k} \sum_{a_{ks} \in E_k^c} P(a_{cu}, a_{ks}) \log \frac{P(a_{cu}, a_{ks})}{P(a_{cu})P(a_{ks})} \tag{4.55}$$

and

$$H(X_k^c, C^k) = - \sum_{a_{cu} \in E_c^k} \sum_{a_{ks} \in E_k^c} P(a_{cu}, a_{ks}) \log P(a_{cu}, a_{ks}). \tag{4.56}$$

The interdependence redundancy measure has a chi-squared distribution:

$$I(X_k^c, C^k) \frac{\chi^2_{df}}{2|S|H(x_k^c, C^k)} \tag{4.57}$$

where *df* is the corresponding degree of freedom having the value $(|E_c^k|-1)(|E_k^c|-1)$. A chi-squared test is then used to select interdependent variables in X at a presumed significant level.

The cluster regrouping process uses an information measure to regroup data iteratively. Wong et al. have proposed an information measure called *normalized surprisal* (NS) to indicate significance of joint information. Using this measure, the information conditioned by an observed event x_k is weighted according to $R(X_k^c, C^K)$, their measure of interdependency with the cluster label variable. Therefore, the higher the interdependency of a conditioning event, the more relevant the event is. NS measures the joint information of a hypothesized value based on the selected set of significant components. It is defined as

$$NS(a_{cj}|x'(a_{cj})) = \frac{I(a_{cj}|x'(a_{cj}))}{m\left(\sum_{k=1}^{m} R(X_k^c, C^k)\right)} \tag{4.58}$$

where $I(a_{cj}|x'(a_{cj}))$ is the summation of the weighted conditional information defined on the incomplete probability distribution scheme as

$$\begin{aligned} I(a_{cj}|x'(a_{cj})) &= \sum_{k=1}^{m} R(X_k^c, C^k) I(a_{cj}|x_k)) \\ &= \sum_{k=1}^{m} R(X_k^c, C^k)\left(-log\ \frac{P(a_{cj}|x_k)}{\sum_{a_{cu}\in E_c^k} P(a_{cu}|x_k)}\right) \end{aligned} \tag{4.59}$$

In rendering a meaningful calculation in the incomplete probability scheme formulation, x_k is selected if

$$\sum_{a_{cu}\in E_c^k} P(a_{cu}|x_k) > T \tag{4.60}$$

where $T \geq 0$ is a size threshold for meaningful estimation. NS can be used in a decision rule in the regrouping process. Let $C = \{a_{c1}, \ldots, a_{cq}\}$ be the set of possible cluster labels. We assign a_{cj} to x_e if

$$NS(a_{cj}|x'(a_{cj})) = \min_{a_{cu}\in C} NS(a_{cu}|x'(a_{cu})).$$

If no component is selected with respect to all hypothesized cluster labels, or if there is more than one label associated with the same minimum NS, then the sample is assigned a dummy label, indicating that the estimated cluster label is still uncertain. Also, zero probability may be encountered in the probability estimation, an unbiased probability based on *Entropy minimax*. In the regrouping algorithm, the cluster label for each sample is estimated iteratively until a stable set of label assignments is attained.

Once the clusters are stable, we take the examples with MVs. Now we use the distance $D(x_i, S) = \min_{x_j \in S_{x_i \neq x_j}} d(x_i, x_j)$ to find the nearest cluster C_i to such instance. From this cluster we compute the centroid x' such that

$$D(x', C_i) < D(x_i, C_i) \tag{4.61}$$

for all instances x_i of the cluster C_i. Once the centroid is obtained, the MV of the example is imputed with the value of the proper attribute of x_i.

4.5.7 Singular Value Decomposition Imputation (SVDI)

In this method, we employ singular value decomposition (4.62) to obtain a set of mutually orthogonal expression patterns that can be linearly combined to approximate the values of all attributes in the data set [93]. These patterns, which in this case are identical to the principle components of the data values' matrix, are further referred to as eigenvalues.

$$A_{m \times m} = U_{m \times m} \Sigma_{m \times n} V^T_{n \times n}. \tag{4.62}$$

Matrix V^T now contains eigenvalues, whose contribution to the expression in the eigenspace is quantified by corresponding eigenvalues on the diagonal of matrix Σ. We then identify the most significant eigenvalues by sorting the eigenvalues based on their corresponding eigenvalue. Although it has been shown that several significant eigenvalues are sufficient to describe most of the expression data, the exact fraction of eigenvalues best for estimation needs to be determined empirically.

Once k most significant eigenvalues from V^T are selected, we estimate a MV j in example i by first regressing this attribute value against the k eigenvalues and then use the coefficients of the regression to reconstruct j from a linear combination of the k eigenvalues. The jth value of example i and the jth values of the k eigenvalues are not used in determining these regression coefficients. It should be noted that SVD can only be performed on complete matrices; therefore we originally substitute row average for all MVs in matrix A, obtaining A'. We then utilize an Regularized EM method to arrive at the final estimate, as follows. Each MV in A' is estimated using the above algorithm, and then the procedure is repeated on the newly obtained matrix, until the total change in the matrix falls below the empirically determined (by the authors [93]) threshold of 0.01 (noted as *stagnation tolerance* in the *EM* algorithm). The other parameters of the *EM* algorithm are the same for both algorithms.

4.5.8 Local Least Squares Imputation (LLSI)

In this method proposed in [49] a target instance that has MVs is represented as a linear combination of similar instances. Rather than using all available instances in the data, only similar instances based on a similarity measure are used, and for that reason the method has the “local” connotation. There are two steps in the LLSI. The first step is to select k instances by the L_2-norm. The second step is regression and estimation, regardless of how the k instances are selected. A heuristic k parameter selection method is used by the authors.

Throughout the section, we will use $X \in \mathbb{R}^{m\times n}$ to denote a dataset with m attributes and n instances. Since LLSI was proposed for microarrays, it is assumed that $m \gg n$. In the data set X, a row $x_i^T \in \mathbb{R}^{1\times n}$ represents expressions of the ith instance in n examples:

$$X = \begin{pmatrix} x_1^T \\ \vdots \\ x_m^T \end{pmatrix} \in \mathbb{R}^{m\times n}$$

A MV in the lth location of the ith instance is denoted as α, i.e.

$$X(i,l) = \mathbf{x}_i(l) = \alpha$$

For simplicity we first assume assuming there is a MV in the first position of the first instance, i.e.

$$X(1,1) = \mathbf{x}_1(1) = \alpha.$$

4.5.8.1 Selecting the Instances

To recover a MV α in the first location $\mathbf{x}_1(1)$ of $\mathbf{x}_1$ in $X \in \mathbb{R}^{m\times n}$, the KNN instance vectors for $\mathbf{x}_1$,

$$\mathbf{x}_{S_i}^T \in \mathbb{R}^{1\times n}, \quad 1 \le i \le k,$$

are found for LLSimpute based on the L_2-norm (LLSimpute). In this process of finding the similar instances, the first component of each instance is ignored due to the fact that $\mathbf{x}_1(1)$ is missing. The LLSimpute based on the Pearson’s correlation coefficient to select the k instances can be consulted in [49].

4.5.8.2 Local Least Squares Imputation

As imputation can be performed regardless of how the k-instances are selected, we present only the imputation based on L_2-norm for simplicity. Based on these k-neighboring instance vectors, the matrix $A \in \mathbb{R}^{k\times(n-1)}$ and the two vectors $\mathbf{b} \in \mathbb{R}^{k\times 1}$ and $\mathbf{w} \in \mathbb{R}^{(n-1)\times 1}$ are formed. The k rows of the matrix A consist of the KNN instances $\mathbf{x}_{S_i}^T \in \mathbb{R}^{1\times n}$, $1 \le i \le k$, with their first values deleted, the elements of the vector b consists of the first components of the k vectors $\mathbf{x}_{S_i}^T$, and the elements of the vector $\mathbf{w}$ are the $n-1$ elements of the instance vector $\mathbf{x}_1$ whose missing first item is deleted. After the matrix A, and the vectors $\mathbf{b}$ and $\mathbf{w}$ are formed, the least squares problem is formulated as

$$\min_{x} ||A^T\mathbf{z} - \mathbf{w}||_2 \tag{4.63}$$

Then, the MV α is estimated as a linear combination of first values of instances

$$\alpha = \mathbf{b}^T\mathbf{z} = \mathbf{b}^T(A^T)^{\dagger}\mathbf{w}, \tag{4.64}$$

where $(A^T)^{\dagger}$ is the pseudoinverse of A^T.

For example, assume that the target instance $\mathbf{x}_1$ has a MV in the first position among a total of six examples. If the MV is to be estimated by the k similar instances, the matrix A, and vectors $\mathbf{b}$ and $\mathbf{w}$ are constructed as

$$\begin{aligned}\begin{pmatrix} x_1^T \\ \vdots \\ x_m^T \end{pmatrix} &= \begin{pmatrix} \alpha & \mathbf{w}^T \\ \mathbf{b} & A \end{pmatrix} \\ &= \begin{pmatrix} \alpha & \mathbf{w}_1 & \mathbf{w}_2 & \mathbf{w}_3 & \mathbf{w}_4 & \mathbf{w}_5 \\ \mathbf{b}_1 & A_{1,1} & A_{1,2} & A_{1,3} & A_{1,4} & A_{1,5} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{b}_k & A_{k,1} & A_{k,2} & A_{k,3} & A_{k,4} & A_{k,5} \end{pmatrix}\end{aligned}$$

where α is the MV and $\mathbf{x}_{S_1}^T, \ldots, \mathbf{x}_{S_k}^T$ are instances similar to $\mathbf{x}_1^T$. From the second to the last components of the neighbor instances, $a_i^T,\ 1 \le i \le k$, form the ith row vector of the matrix A. The vector w of the known elements of target instance $\mathbf{x}_1$ can be represented as a linear combination

$$\mathbf{w} \simeq \mathbf{z}_1\mathbf{a}_1 + \mathbf{z}_2\mathbf{a}_2 + \cdots + \mathbf{z}_k\mathbf{a}_k$$

where $\mathbf{z}_i$ are the coefficients of the linear combination, found from the least squares formulation (4.63). Accordingly, the MV α in $\mathbf{x}_1$ can be estimated by

$$\alpha = \mathbf{b}^T x = \mathbf{b}_1\mathbf{z}_1 + \mathbf{b}_2\mathbf{z}_2 + \cdots + \mathbf{b}_k\mathbf{z}_k$$

Now, we deal with the case in which there is more than one MV in a instance vector. In this case, to recover the total of q MVs in any of the locations of the instance $\mathbf{x}_1$, first, the KNN instance vectors for x_1,

$$\mathbf{x}_{S_i}^T \in \mathbb{R}^{1\times n}, \quad 1 \le i \le k,$$

are found. In this process of finding the similar instances, the q components of each instance at the q locations of MVs in $\mathbf{x}_1$ are ignored. Then, based on these

k neighboring instance vectors, a matrix $A \in \mathbb{R}^{k\times(n-q)}$ a matrix $B \in \mathbb{R}^{k\times q}$ and a vector $\mathbf{w} \in \mathbb{R}^{(n-q)\times 1}$ are formed. The ith row vector $\mathbf{a}_i^T$ of the matrix A consists of the ith nearest neighbor instances $\mathbf{x}_{S_i}^T \in \mathbb{R}^{1\times n}$, $1 \le i \le k$, with its elements at the q missing locations of MVs of $\mathbf{x}_1$ excluded. Each column vector of the matrix B consists of the values of the jth location of the MVs ($1 \le j \le q$) of the k vectors $\mathbf{x}_{S_i}^T$. The elements of the vector $\mathbf{w}$ are the $n-q$ elements of the instance vector $\mathbf{x}$ whose missing items are deleted. After the matrices A and B and a vector $\mathbf{w}$ are formed, the least squares problem is formulated as

$$\min_x ||A^T\mathbf{z} - \mathbf{w}||_2 \tag{4.65}$$

Then, the vector $\mathbf{u} = (\alpha_1, \alpha_2, \ldots, \alpha_q)^T$ of q MVs can be estimated as

$$\mathbf{u} = \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_q \end{pmatrix} = B^T\mathbf{z} = B^T(A^T)^\dagger\mathbf{w}, \tag{4.66}$$

where $(A^T)^\dagger$ is the pseudoinverse of A^T.

Table 4.1 Recent and most well-known imputation methods involving ML techniques

Clustering		Kernel methods	
MLP hybrid	[4]	Mixture-kernel-based iterative estimator	[105]
Rough fuzzy subspace clustering	[89]	*Nearest neighbors*	
LLS based	[47]	ICkNNI	[40]
Fuzzy c-means with SVR and Gas	[3]	Iterative KNNI	[101]
Biclustering based	[32]	CGImpute	[22]
KNN based	[46]	Boostrap for maximum likelihood	[72]
Hierarchical Clustering	[30]	kDMI	[75]
K2 clustering	[39]	*Ensembles*	
Weighted K-means	[65]	Random Forest	[42]
Gaussian mixture clustering	[63]	Decision forest	[76]
ANNs		Group Method of Data Handling (GMDH)	[104]
RBFN based	[90]	Boostrap	[56]
Wavelet ANNs	[64]	*Similarity and correlation*	
Multi layer perceptron	[88]	FIMUS	[77]
ANNs framework	[34]	*Parameter estimation for regression imputation*	
Self-organizing maps	[58]	EAs for covariance matrix estimation	[31]
Generative Topographic Mapping	[95]	Iterative mutual information imputation	[102]
Bayesian networks		CMVE	[87]
Dynamic bayesian networks	[11]	DMI (EM + decision trees)	[74]
Bayesian networks with weights	[60]	WLLSI	[12]

4.5.9 Recent Machine Learning Approaches to Missing Values Imputation

Although we have tried to provide an extensive introduction to the most used and basic imputation methods based on ML techniques, there is a great amount of journal publications showing their application and particularization to real world problems. We would like to give the reader a summarization of the latest and more important imputation methods presented at the current date of publication, both extensions of the introduced ones and completely novel ones in Table 4.1.

4.6 Experimental Comparative Analysis

In this section we aim to provide the reader with a general overview of the behavior and properties of all the imputation methods presented above. However, this is not an easy task. The main question is: what is a good imputation method?

As multiple imputation is a very resource consuming approach, we will focus on the single imputation methods described in this chapter.

4.6.1 Effect of the Imputation Methods in the Attributes' Relationships

From an unsupervised data point of view, those imputation methods able to generate values close to the true but unknown MV should be the best. This idea has been explored in the literature by means of using complete data sets and then artificially introducing MVs. Please note that such a mechanism will act as a MCAR MV generator mechanism, validating the use of imputation methods. Then, imputation methods are applied to the data and an estimation of how far is the estimation to the original (and known) value. Authors usually choose the mean square error (MSE) or root mean square error (RMSE) to quantify and compare the imputation methods over a set of data sets [6, 32, 41, 77].

On the other hand, other problems arise when we do not have the original values or the problem is supervised. In classification, for example, it is more demanding to impute values that will constitute an easier and more generalizable problem. As a consequence in this paradigm a good imputation method will enable the classifier to obtain better accuracy. This is harder to measure, as we are relating two different values: the MV itself and the class label assigned to the example. Neither MSE or RMSE can provide us with such kind of information.

One way to measure how good the imputation is for the supervised task is to use Wilson's Noise Ratio. This measure proposed by [98] observes the noise in the data set. For each instance of interest, the method looks for the KNN (using the

Euclidean distance), and uses the class labels of such neighbors in order to classify the considered instance. If the instance is not correctly classified, then the variable *noise* is increased by one unit. Therefore, the final noise ratio will be

$$\text{Wilson's Noise} = \frac{noise}{\text{\#instances in the data set}}$$

After imputing a data set with different imputation methods, we can measure how disturbing the imputation method is for the classification task. Thus by using Wilson's noise ratio we can observe which imputation methods reduce the impact of the MVs as a noise, and which methods produce noise when imputing.

Another approach is to use the MI (MI) which is considered to be a good indicator of relevance between two random variables [18]. Recently, the use of the MI measure in FS has become well-known and seen to be successful [51, 52, 66]. The use of the MuI measure for continuous attributes has been tackled by [51], allowing us to compute the Mui measure not only in nominal-valued data sets.

In our approach, we calculate the Mui between each input attribute and the class attribute, obtaining a set of values, one for each input attribute. In the next step we compute the ratio between each one of these values, considering the imputation of the data set with one imputation method in respect to the not imputed data set. The average of these ratios will show us if the imputation of the data set produces a gain in information:

$$\text{Avg. Mui Ratio} = \frac{\sum_{x_i \in X} \frac{Mui_\alpha(x_i)+1}{Mui(x_i)+1}}{|X|}$$

where X is the set of input attributes, $Mui_\alpha(i)$ represents the Mui value of the ith attribute in the imputed data set and $Mui(i)$ is the Mui value of the ith input attribute in the not imputed data set. We have also applied the Laplace correction, summing 1 to both numerator and denominator, as an Mui value of zero is possible for some input attributes.

The calculation of $Mui(x_i)$ depends on the type of attribute x_i. If the attribute x_i is nominal, the Mui between x_i and the class label Y is computed as follows:

$$Mui_{nominal}(x_i) = I(x_i; Y) = \sum_{z \in x_i} \sum_{y \in Y} p(z, y) log_2 \frac{p(z, y)}{p(z)p(y)}.$$

On the other hand, if the attribute x_i is numeric, we have used the Parzen window density estimate as shown in [51] considering a Gaussian window function:

$$Mui_{numeric}(x_i) = I(x_i; Y) = H(Y) - H(C|X);$$

where $H(Y)$ is the entropy of the class label

$$H(Y) = -\sum_{y \in Y} p(y) log_2 p(y);$$

and $H(C|X)$ is the conditional entropy

$$H(Y|x_i) = -\sum_{z \in x_i} \sum_{y \in Y} p(z, y) log_2 p(y|z).$$

Considering each sample has the same probability, applying the Bayesian rule and approximating $p(y|z)$ by the Parzen window we get:

$$\hat{H}(Y|x_i) = -\sum_{j=1}^{n} \frac{1}{n} \sum_{y=1}^{N} \hat{p}(y|z_j) log_2 \hat{p}(y|z_j)$$

where n is the number of instances in the data set, N is the total number of class labels and $\hat{p}(c|x)$ is

$$\hat{p}(y|z) = \frac{\sum_{i \in I_c} exp\left(-\frac{(z-z_i)\Sigma^{-1}(z-z_i)}{2h^2}\right)}{\sum_{k=1}^{N} \sum_{i \in I_k} exp\left(-\frac{(z-z_i)\Sigma^{-1}(z-z_i)}{2h^2}\right)}.$$

In this case, I_c is the set of indices of the training examples belonging to class c, and Σ is the covariance of the random variable $(z - z_i)$.

Let us consider all the single imputation methods presented in this chapter. For the sake of simplicity we will omit the Multiple Imputation approaches, as it will require us to select a probability model for all the data sets, which would be infeasible. In Table 4.2 we have summarized the Wilson's noise ratio values for 21 data sets with MVs from those presented in Sect. 2.1. We must point out that the results of Wilson's noise ratio and Mui are related to a given data set. Hence, the characteristics of the proper data appear to determine the values of this measure.

Looking at the results from Table 4.2 we can observe which imputation methods reduce the impact of the MVs as noise, and which methods produce noise when imputing. In addition the MI ratio allows us to relate the attributes to the imputation results. A value of the Mui ratio higher than 1 will indicate that the imputation is capable of relating more of the attributes individually to the class labels. A value lower than 1 will indicate that the imputation method is adversely affecting the relationship between the individual attributes and the class label.

If we consider the average Mui ratio in Table 4.2 we can observe that the average ratios are usually close to 1; that is, the use of imputation methods appears to harm the relationship between the class label and the input attribute little or not at all, even improving it in some cases. However, the MI considers only one attribute at a time and therefore the relationships between the input attributes are ignored. The imputation

Table 4.2 Wilson's noise ratio values and average MI values per data set

Data set	Imp. Method	% Wilson's Noise	Avg. Mui ratio	Data set	Imp. Method	% Wilson's Noise	Avg. Mui ratio	Data set	Imp. Method	% Wilson's Noise	Avg. Mui ratio
CLE	MC	50.0000	0.998195	HOV	MC	7.9208	0.961834	HEP	MC	17.3333	0.963765
	CMC	50.0000	0.998585		CMC	5.4455	1.105778		CMC	16.0000	0.990694
	KNNI	50.0000	0.998755		KNNI	7.4257	0.965069		KNNI	20.0000	0.978564
	WKNNI	50.0000	0.998795		WKNNI	7.4257	0.965069		WKNNI	20.0000	0.978343
	KMI	50.0000	0.998798		KMI	7.4257	0.961525		KMI	20.0000	0.980094
	FKMI	50.0000	0.998889		FKMI	7.9208	0.961834		FKMI	17.3333	0.963476
	SVMI	50.0000	0.998365		SVMI	6.9307	0.908067		SVMI	17.3333	1.006819
	EM	66.6667	0.998152		EM	11.8812	0.891668		EM	22.6667	0.974433
	SVDI	66.6667	0.997152		SVDI	8.9109	0.850361		SVDI	21.3333	0.967673
	BPCA	50.0000	0.998701		BPCA	6.9307	1.091675		BPCA	21.3333	0.994420
	LLSI	50.0000	0.998882		LLSI	**4.9505**	**1.122904**		LLSI	18.6667	0.995464
	EC	**33.3333**	**1.000148**		EC	7.4257	1.007843		EC	**16.0000**	**1.024019**
WIS	MC	18.7500	0.999004	WAT	MC	31.5068	0.959488	MUS	MC	**0.0000**	1.018382
	CMC	**12.5000**	0.999861		CMC	**21.2329**	0.967967		CMC	**0.0000**	1.018382
	KNNI	**12.5000**	0.999205		KNNI	27.3973	0.961601		KNNI	**0.0000**	0.981261
	WKNNI	**12.5000**	0.999205		WKNNI	27.3973	0.961574		WKNNI	**0.0000**	0.981261
	KMI	**12.5000**	0.999322		KMI	27.3973	0.961361		KMI	**0.0000**	1.018382
	FKMI	**12.5000**	0.998923		FKMI	31.5068	0.961590		FKMI	**0.0000**	1.018382
	SVMI	**12.5000**	0.999412		SVMI	23.9726	0.967356		SVMI	**0.0000**	0.981261
	EM	**12.5000**	0.990030		EM	46.5753	0.933846		EM	**0.0000**	**1.142177**
	SVDI	**12.5000**	0.987066		SVDI	49.3151	0.933040		SVDI	**0.0000**	1.137152
	BPCA	**12.5000**	0.998951		BPCA	26.0274	0.964255		BPCA	**0.0000**	0.987472

(continued)

Table 4.2 (continued)

Data set	Imp.	% Wilson's	Avg. Mui	Data set	Imp.	% Wilson's	Avg. Mui	Data set	Imp.	% Wilson's	Avg. Mui
	Method	Noise	ratio		Method	Noise	ratio		Method	Noise	ratio
	LLSI	**12.5000**	0.999580		LLSI	25.3425	0.964063		LLSI	**0.0000**	0.977275
	EC	**12.5000**	**1.000030**		EC	22.6027	**1.027369**		EC	**0.0000**	1.017366
CRX	MC	18.9189	1.000883	SPO	MC	27.2727	0.997675	POS	MC	**33.3333**	1.012293
	CMC	18.9189	1.000966		CMC	**22.7273**	**1.022247**		CMC	**33.3333**	1.012293
	KNNI	21.6216	0.998823		KNNI	27.2727	0.999041		KNNI	**33.3333**	1.012293
	WKNNI	21.6216	0.998870		WKNNI	27.2727	0.999041		WKNNI	**33.3333**	1.012293
	KMI	21.6216	1.001760		KMI	27.2727	0.998464		KMI	**33.3333**	1.012293
	FKMI	18.9189	1.000637		FKMI	27.2727	0.997675		FKMI	**33.3333**	1.012293
	SVMI	13.5135	0.981878		SVMI	27.2727	1.015835		SVMI	**33.3333**	1.012293
	EM	32.4324	0.985609		EM	36.3636	0.982325		EM	**33.3333**	1.012293
	SVDI	27.0270	0.976398		SVDI	31.8182	0.979187		SVDI	**33.3333**	1.014698
	BPCA	21.6216	0.999934		BPCA	27.2727	1.006236		BPCA	**33.3333**	1.012293
	LLSI	18.9189	1.001594		LLSI	27.2727	1.004821		LLSI	**33.3333**	**1.018007**
	EC	**13.5135**	**1.008718**		EC	27.2727	1.018620		EC	**33.3333**	0.997034
BRE	MC	55.5556	0.998709	BAN	MC	25.4753	1.012922	ECH	MC	40.0000	0.981673
	CMC	55.5556	0.998709		CMC	24.3346	1.070857		CMC	40.0000	0.995886
	KNNI	55.5556	0.992184		KNNI	23.1939	0.940369		KNNI	46.6667	0.997912
	WKNNI	55.5556	0.992184		WKNNI	22.8137	0.940469		WKNNI	44.4444	**0.998134**
	KMI	55.5556	0.998709		KMI	25.4753	1.016101		KMI	46.6667	0.967169
	FKMI	55.5556	0.998709		FKMI	24.3346	1.020989		FKMI	40.0000	0.983606
	SVMI	55.5556	0.998709		SVMI	**21.2928**	**1.542536**		SVMI	44.4444	0.987678
	EM	**44.4444**	**1.013758**		EM	26.2357	1.350315		EM	51.1111	0.967861
	SVDI	**44.4444**	0.999089		SVDI	22.4335	1.365572		SVDI	48.8889	0.935855

Table 4.2 (continued)

Data set	Imp.	% Wilson's	Avg. Mui	Data set	Imp.	% Wilson's	Avg. Mui	Data set	Imp.	% Wilson's	Avg. Mui
	Method	Noise	ratio		Method	Noise	ratio		Method	Noise	ratio
	BPCA	66.6667	1.000201		BPCA	23.9544	1.010596		BPCA	44.4444	0.972327
	LLSI	66.6667	1.000201		LLSI	24.7148	1.015033		LLSI	**37.7778**	0.988591
	EC	66.6667	1.001143		EC	23.5741	1.102328		EC	48.8889	0.970029
AUT	MC	45.6522	0.985610	HOC	MC	19.3906	0.848649	SOY	MC	**2.4390**	1.056652
	CMC	41.3043	0.991113		CMC	**10.2493**	**2.039992**		CMC	**2.4390**	1.123636
	KNNI	41.3043	0.986239		KNNI	20.2216	0.834734		KNNI	**2.4390**	1.115818
	WKNNI	41.3043	0.985953		WKNNI	19.1136	0.833982		WKNNI	**2.4390**	1.115818
	KMI	41.3043	0.985602		KMI	21.8837	0.821936		KMI	**2.4390**	1.056652
	FKMI	45.6522	0.984694		FKMI	20.4986	0.849141		FKMI	**2.4390**	1.056652
	SVMI	43.4783	0.991850		SVMI	20.2216	0.843456		SVMI	**2.4390**	**1.772589**
	EM	58.6957	0.970557		EM	21.0526	0.775773		EM	**2.4390**	1.099286
	SVDI	52.1739	0.968938		SVDI	21.0526	0.750930		SVDI	7.3171	1.065865
	BPCA	43.4783	0.986631		BPCA	19.3906	0.964587		BPCA	7.3171	1.121603
	LLSI	45.6522	0.985362		LLSI	20.4986	0.926068		LLSI	**2.4390**	1.159610
	EC	**30.4348**	**1.007652**		EC	20.7756	0.911543		EC	**2.4390**	1.222631
PRT	MC	71.0145	0.949896	AUD	MC	38.7387	0.990711	MAM	MC	21.3740	0.974436
	CMC	**60.8696**	**1.120006**		CMC	**32.8829**	1.032162		CMC	**13.7405**	1.029154
	KNNI	69.5652	0.976351		KNNI	38.7387	0.993246		KNNI	25.9542	0.965926
	WKNNI	69.5652	0.976351		WKNNI	38.7387	0.993246		WKNNI	25.9542	0.965926
	KMI	71.0145	0.949896		KMI	38.7387	1.000235		KMI	24.4275	0.966885
	FKMI	71.0145	0.949896		FKMI	38.7387	0.990711		FKMI	20.6107	0.974228
	SVMI	68.1159	1.038152		SVMI	37.8378	1.007958		SVMI	16.7939	**1.272993**

(continued)

Table 4.2 (continued)

Data set	Imp.	% Wilson's	Avg. Mui	Data set	Imp.	% Wilson's	Avg. Mui	Data set	Imp.	% Wilson's	Avg. Mui
	Method	Noise	ratio		Method	Noise	ratio		Method	Noise	ratio
	EM	88.4058	0.461600		EM	53.6036	1.129168		EM	20.6107	0.980865
	SVDI	91.7874	0.485682		SVDI	46.3964	1.065091		SVDI	27.4809	1.052790
	BPCA	71.4976	0.987598		BPCA	40.5405	1.156676		BPCA	25.1908	0.978209
	LLSI	69.5652	1.016230		LLSI	36.9369	1.061197		LLSI	26.7176	0.994349
	EC	66.1836	1.053185		EC	37.8378	**1.209608**		EC	18.3206	1.269505
DER	MC	**0.0000**	1.000581	LUN	MC	80.0000	0.996176	OZO	MC	4.8035	0.982873
	CMC	**0.0000**	**1.002406**		CMC	80.0000	1.008333		CMC	**3.6390**	**0.989156**
	KNNI	**0.0000**	0.999734		KNNI	80.0000	0.996176		KNNI	4.3668	0.982759
	WKNNI	**0.0000**	0.999734		WKNNI	80.0000	0.996176		WKNNI	4.5124	0.982721
	KMI	**0.0000**	1.000581		KMI	80.0000	0.996176		KMI	4.9491	0.982495
	FKMI	**0.0000**	1.000581		FKMI	80.0000	0.996176		FKMI	4.0757	0.982951
	SVMI	**0.0000**	1.001566		SVMI	80.0000	1.006028		SVMI	3.7846	0.988297
	EM	**0.0000**	1.000016		EM	**20.0000**	1.067844		EM	4.8035	0.979977
	SVDI	**0.0000**	0.999691		SVDI	40.0000	**1.076334**		SVDI	4.8035	0.979958
	BPCA	**0.0000**	0.999633		BPCA	80.0000	0.996447		BPCA	4.3668	0.983318
	LLSI	**0.0000**	0.999170		LLSI	80.0000	1.007612		LLSI	4.2213	0.983508
	EC	**0.0000**	1.000539		EC	80.0000	1.002385		EC	4.8035	0.944747

methods estimate the MVs using such relationships and can afford improvements in the performance of the classifiers. Hence the highest values of average Mui ratios could be related to those methods which can obtain better estimates for the MVs, and maintaining the relationship degree between the class labels and the isolated input attributes. It is interesting to note that when analyzing the Mui ratio, the values do not appear to be as highly data dependant as Wilson's noise ratio, as the values for all the data sets are more or less close to each other.

If we count the methods with the lowest Wilson's noise ratios in each data set in Table 4.2, we find that the CMC method is first, with 12 times being the lowest one, and the EC method is second with 9 times being the lowest one. If we count the methods with the highest MI ratio in each data set, the EC method has the highest ratio for 7 data sets and is therefore the first one. The CMC method has the highest ratio for 5 data sets and is the second one in this case. Immediately the next question arises: are these methods also the best for the performance of the learning methods applied afterwards? We try to answer this question in the following.

4.6.2 Best Imputation Methods for Classification Methods

Our aim is to use the same imputation results as data sets used in the previous Sect. 4.6.1 as the input for a series of well known classifiers in order to shed light on the question "which is the best imputation method?". Let us consider a wide range of classifiers grouped by their nature, as that will help us to limit the comparisons needed to be made. We have grouped them in three sub-categories. In Table 4.3 we summarize the classification methods we have used, organized in these three categories. The description of the former categories is as follows:

- The first group is the *Rule Induction Learning* category. This group refers to algorithms which infer rules using different strategies.
- The second group represents the *Black Box Methods*. It includes ANNs, SVMs and statistical learning.
- The third and last group corresponds to the *Lazy Learning* (LL) category. This group incorporates methods which do not create any model, but use the training data to perform the classification directly.

Some methods do not work with numerical attributes (CN2, AQ and Naïve-Bayes). In order to discretize the numerical values, we have used the well-known discretizer proposed by [28]. For the SVM methods (C-SVM, ν-SVM and SMO), we have applied the usual preprocessing in the literature to these methods [25]. This preprocessing consists of normalizing the numerical attributes to the [0, 1] range, and binarizing the nominal attributes. Some of the presented classification methods in the previous section have their own MVs treatment that will trigger when no imputation is made (DNI): C4.5 uses a probabilistic approach to handling MVs and CN2 applies the MC method by default in these cases. For ANNs [24] proposed to replace MVs with zero so as not to trigger the corresponding neuron which the MV is applied to.

Table 4.3 Classifiers used by categories

Method	Acronym	References
Rule Induction Learning		
C4.5	C4.5	[73]
Ripper	Ripper	[16]
CN2	CN2	[14]
AQ-15	AQ	[59]
PART	PART	[33]
Slipper	Slipper	[15]
Scalable Rule Induction Induction	SRI	[68]
Rule Induction Two In One	Ritio	[100]
Rule Extraction System version 6	Rule-6	[67]
Black Box Methods		
Multi-Layer Perceptron	MLP	[61]
C-SVM	C-SVM	[25]
ν-SVM	ν-SVM	[25]
Sequential Minimal Optimization	SMO	[70]
Radial Basis Function Network	RBFN	[8]
RBFN Decremental	RBFND	[8]
RBFN Incremental	RBFNI	[69]
Logistic	LOG	[10]
Naïve-Bayes	NB	[21]
Learning Vector Quantization	LVQ	[7]
Lazy Learning		
1-NN	1-NN	[57]
3-NN	3-NN	[57]
Locally Weighted Learning	LWL	[2]
Lazy Learning of Bayesian Rules	LBR	[103]

As shown here all the detailed accuracy values for each fold, data set, imputation method and classifier would be too long, we have used Wilcoxon's Signed Rank test to summarize them. For each classifier, we have compared every imputation method along with the rest in pairs. Every time the classifier obtains a better accuracy value for an imputation method than another one and the statistical test yield a $p-value < 0.1$ we count it as a win for the former imputation method. In another case it is a tie when $p-value > 0.1$.

In the case of rule induction learning in Table 4.4 we show the average ranking or each imputation method for every classifier belonging to this group. We can observe that, for the rule induction learning classifiers, the imputation methods FKMI, SVMI and EC perform best. The differences between these three methods in average rankings are low. Thus we can consider that these three imputation methods are the most suitable for this kind of classifier. They are well separated from the other

Table 4.4 Average ranks for the Rule Induction Learning methods

	C45	Ripper	PART	Slipper	AQ	CN2	SRI	Ritio	Rules-6	Avg.	Ranks
IM	5	8.5	1	4	6.5	10	6.5	6	5	5.83	4
EC	2.5	8.5	6.5	1	6.5	5.5	6.5	6	1	4.89	3
KNNI	9	2.5	6.5	11	11	5.5	11.5	11	11	8.78	11
WKNNI	11	2.5	6.5	7	6.5	1	11.5	6	11	7.00	8
KMI	5	2.5	6.5	3	6.5	5.5	9.5	12	7.5	6.44	6
FKMI	7.5	2.5	6.5	10	2	5.5	1	2	3	4.44	1
SVMI	1	5.5	6.5	7	1	5.5	6.5	6	2	4.56	2
EM	13	12	6.5	7	12	13	3	6	4	8.50	10
SVDI	11	11	6.5	12	10	12	9.5	10	11	10.33	12
BPCA	14	13	13	7	13	14	13	13	13	12.56	14
LLSI	11	5.5	6.5	7	6.5	11	3	6	7.5	7.11	9
MC	7.5	8.5	6.5	2	6.5	5.5	3	6	7.5	5.89	5
CMC	5	8.5	12	13	3	5.5	6.5	1	7.5	6.89	7
DNI	2.5	14	14	14	14	5.5	14	14	14	11.78	13

Table 4.5 Average ranks for the Black Box methods

	RBFN	RBFND	RBFNI	LOG	LVQ	MLP	NB	ν-SVM	C-SVM	SMO	Avg.	Ranks
IM	9	6.5	4.5	6	3.5	13	12	10	5.5	5.5	7.55	10
EC	1	1	1	3	7	8.5	10	13	1	2	4.75	1
KNNI	5	6.5	10.5	9	7	11	6.5	8	5.5	5.5	7.45	9
WKNNI	13	6.5	4.5	10	10	4.5	6.5	4.5	5.5	5.5	7.05	6
KMI	3.5	2	7	3	11	3	4.5	8	5.5	9	5.65	2
FKMI	12	6.5	10.5	3	1.5	4.5	11	4.5	5.5	3	6.20	3
SVMI	2	11.5	2.5	7.5	3.5	1.5	13	8	11	9	6.95	5
EM	3.5	6.5	13	12	12.5	10	4.5	4.5	10	11.5	8.80	11
SVDI	9	6.5	7	11	12.5	8.5	3	11.5	12	11.5	9.25	12
BPCA	14	14	14	13	7	14	2	2	13	13	10.60	14
LLSI	6	6.5	10.5	7.5	7	6.5	9	4.5	5.5	9	7.20	7
MC	9	6.5	10.5	3	7	6.5	8	11.5	5.5	5.5	7.30	8
CMC	9	13	2.5	3	1.5	1.5	14	14	5.5	1	6.50	4
DNI	9	11.5	7	14	14	12	1	1	14	14	9.75	13

imputation methods and we cannot choose the best method from among these three. On the other hand, BPCA and DNI are the worst methods.

In Table 4.5 we can observe the rankings associated with the methods belonging to the black-boxes modeling category. As can be appreciated, for black-boxes modelling the differences between imputation methods are even more evident. We can select the EC method as the best solution, as it has a difference of ranking of almost 1 with KMI, which stands as the second best. This difference increases when considering

Table 4.6 Average ranks for the Lazy Learning methods

	1-NN	3-NN	LBR	LWL	Avg.	Ranks
IM	5	11	5	8	7.25	7
EC	9.5	13	9	8	9.88	12
KNNI	2.5	5.5	9	8	6.25	4
WKNNI	4	5.5	9	8	6.63	5
KMI	12	5.5	9	2.5	7.25	8
FKMI	6	1.5	9	2.5	4.75	3
SVMI	9.5	9	3	8	7.38	9
EM	11	5.5	9	2.5	7.00	6
SVDI	13	12	1	12	9.50	11
BPCA	14	14	13	13	13.50	14
LLSI	7.5	5.5	9	8	7.50	10
MC	7.5	1.5	3	2.5	3.63	1
CMC	1	5.5	3	8	4.38	2
DNI	2.5	10	14	14	10.13	13

the third best, FKMI. No other family of classifiers present this gap in the rankings. Therefore, in this family of classification methods we could, with some confidence, establish the EC method as the best choice. The DNI and IM methods are among the worst. This means that for the black-boxes modelling methods the use of some kind of MV treatment is mandatory, whereas the EC method is the most suitable one. As with the RIL methods, the BPCA method is the worst choice, with the highest ranking.

Finally the results for the last LL group are presented in Table 4.6. For the LL models, the MC method is the best with the lowest average ranking. The CMC method, which is relatively similar to MC, also obtains a low rank very close to MC's. Only the FKMI method obtains a low enough rank to be compared with the MC and CMC methods. The rest of the imputation methods are far from these lowest ranks with almost two points of difference in the ranking. Again, the DNI and IM methods obtain high rankings. The DNI method is one of the worst, with only the BPCA method performing worse. As with the black-boxes modelling models, the imputation methods produce a significant improvement in the accuracy of these classification methods.

4.6.3 Interesting Comments

In this last Section we have carried out an experimental comparison among the imputation methods presented in this chapter. We have tried to obtain the best imputation choice by means of non-parametric statistical testing. The results obtained concur with previous studies:

- The imputation methods which fill in the MVs outperform the case deletion (IM method) and the lack of imputation (DNI method).
- There is no universal imputation method which performs best for all classifiers.

In Sect. 4.6.1 we have analyzed the influence of the imputation methods in the data in respect to two measures. These two measures are the *Wilson's noise ratio* and the *average MI difference*. The first one quantifies the noise induced by the imputation method in the instances which contain MVs. The second one examines the increment or decrement in the relationship of the isolated input attributes with respect to the class label. We have observed that the CMC and EC methods are the ones which introduce less noise and maintain the MI better.

According to the results in Sect. 4.6.2, the particular analysis of the MVs treatment methods conditioned to the classification methods' groups seems necessary. Thus, we can stress the recommended imputation algorithms to be used based on the classification method's type, as in the case of the *FKMI* imputation method for the Rule Induction Learning group, the *EC* method for the black-boxes modelling Models and the *MC* method for the Lazy Learning models. We can confirm the positive effect of the imputation methods and the classifiers' behavior, and the presence of more suitable imputation methods for some particular classifier categories than others.

References

1. Acuna, E., Rodriguez, C.: Classification, Clustering and Data Mining Applications. Springer, Berlin (2004)
2. Atkeson, C.G., Moore, A.W., Schaal, S.: Locally weighted learning. Artif. Intell. Rev. **11**, 11–73 (1997)
3. Aydilek, I.B., Arslan, A.: A hybrid method for imputation of missing values using optimized fuzzy c-means with support vector regression and a genetic algorithm. Inf. Sci. **233**, 25–35 (2013)
4. Azim, S., Aggarwal, S.: Hybrid model for data imputation: using fuzzy c-means and multi layer perceptron. In: Advance Computing Conference (IACC), 2014 IEEE International, pp. 1281–1285 (2014)
5. Barnard, J., Meng, X.: Applications of multiple imputation in medical studies: from aids to nhanes. Stat. Methods Med. Res. **8**(1), 17–36 (1999)
6. Batista, G., Monard, M.: An analysis of four missing data treatment methods for supervised learning. Appl. Artif. Intell. **17**(5), 519–533 (2003)
7. Bezdek, J., Kuncheva, L.: Nearest prototype classifier designs: an experimental study. Int. J. Intell. Syst. **16**(12), 1445–1473 (2001)
8. Broomhead, D., Lowe, D.: Multivariable functional interpolation and adaptive networks. Complex Systems **11**, 321–355 (1988)
9. van Buuren, S., Groothuis-Oudshoorn, K.: MICE: multivariate imputation by chained equations in r. J. Stat. Softw. **45**(3), 1–67 (2011)
10. le Cessie, S., van Houwelingen, J.: Ridge estimators in logistic regression. Appl. Stat. **41**(1), 191–201 (1992)
11. Chai, L., Mohamad, M., Deris, S., Chong, C., Choon, Y., Ibrahim, Z., Omatu, S.: Inferring gene regulatory networks from gene expression data by a dynamic bayesian network-based model. In: Omatu, S., De Paz Santana, J.F., González, S.R., Molina, J.M., Bernardos, A.M.,

Rodríguez, J.M.C. (eds.) Distributed Computing and Artificial Intelligence, Advances in Intelligent and Soft Computing, pp. 379–386. Springer, Berlin (2012)
12. Ching, W.K., Li, L., Tsing, N.K., Tai, C.W., Ng, T.W., Wong, A.S., Cheng, K.W.: A weighted local least squares imputation method for missing value estimation in microarray gene expression data. Int. J. Data Min. Bioinform. **4**(3), 331–347 (2010)
13. Chow, C., Liu, C.: Approximating discrete probability distributions with dependence trees. IEEE Trans. Inf. Theor. **14**(3), 462–467 (1968)
14. Clark, P., Niblett, T.: The CN2 induction algorithm. Machine Learning **3**(4), 261–283 (1989)
15. Cohen, W., Singer, Y.: A simple and fast and effective rule learner. In: Proceedings of the Sixteenth National Conference on Artificial Intelligence, pp. 335–342 (1999)
16. Cohen, W.W.: Fast effective rule induction. In: Proceedings of the Twelfth International Conference on Machine Learning (ICML), pp. 115–123 (1995).
17. Cortes, C., Vapnik, V.: Support vector networks. Machine Learning **20**, 273–297 (1995)
18. Cover, T.M., Thomas, J.A.: Elements of Information Theory, 2 edn. Wiley, New York (1991)
19. Daniel, R.M., Kenward, M.G.: A method for increasing the robustness of multiple imputation. Comput. Stat. Data Anal. **56**(6), 1624–1643 (2012)
20. Dempster, A., Laird, N., Rubin, D.: Maximum likelihood estimation from incomplete data via the EM algorithm (with discussion). J. Roy. Statist. Soc. Ser. B **39**, 1–38 (1977)
21. Domingos, P., Pazzani, M.: On the optimality of the simple bayesian classifier under zero-one loss. Machine Learning **29**, 103–137 (1997)
22. Dorri, F., Azmi, P., Dorri, F.: Missing value imputation in dna microarrays based on conjugate gradient method. Comp. Bio. Med. **42**(2), 222–227 (2012)
23. Dunning, T., Freedman, D.: Modeling section effects, Sage, pp. 225–231 (2008)
24. Ennett, C.M., Frize, M., Walker, C.R.: Influence of missing values on artificial neural network performance. Stud. Health Technol. Inform. **84**, 449–453 (2001)
25. Fan, R.E., Chen, P.H., Lin, C.J.: Working set selection using second order information for training support vector machines. J. Machine Learning Res. **6**, 1889–1918 (2005)
26. Farhangfar, A., Kurgan, L., Dy, J.: Impact of imputation of missing values on classification error for discrete data. Pattern Recognit. **41**(12), 3692–3705 (2008). http://dx.doi.org/10.1016/j.patcog.2008.05.019
27. Farhangfar, A., Kurgan, L.A., Pedrycz, W.: A novel framework for imputation of missing values in databases. IEEE Trans. Syst. Man Cybern. Part A **37**(5), 692–709 (2007)
28. Fayyad, U., Irani, K.: Multi-interval discretization of continuous-valued attributes for classification learning. In: 13th International Joint Conference on Uncertainly in Artificial Intelligence(IJCAI93), pp. 1022–1029 (1993)
29. Feng, H., Guoshun, C., Cheng, Y., Yang, B., Chen, Y.: A SVM regression based approach to filling in missing values. In: Khosla, R., Howlett, R.J., Jain, L.C. (eds.) KES (3), Lecture Notes in Computer Science, vol. 3683, pp. 581–587. Springer, Berlin (2005)
30. Feng, X., Wu, S., Liu, Y.: Imputing missing values for mixed numeric and categorical attributes based on incomplete data hierarchical clustering. In: Proceedings of the 5th International Conference on Knowledge Science, Engineering and Management, KSEM'11, pp. 414–424 (2011)
31. Figueroa García, J.C., Kalenatic, D., Lopez Bello, C.A.: Missing data imputation in multivariate data by evolutionary algorithms. Comput. Hum. Behav. **27**(5), 1468–1474 (2011)
32. de França, F.O., Coelho, G.P., Zuben, F.J.V.: Predicting missing values with biclustering: a coherence-based approach. Pattern Recognit. **46**(5), 1255–1266 (2013)
33. Frank, E., Witten, I.: Generating accurate rule sets without global optimization. In: Proceedings of the 15th International Conference on Machine Learning, pp. 144–151 (1998)
34. Gheyas, I.A., Smith, L.S.: A neural network-based framework for the reconstruction of incomplete data sets. Neurocomputing **73**(16–18), 3039–3065 (2010)
35. Gibert, K.: Mixed intelligent-multivariate missing imputation. Int. J. Comput. Math. **91**(1), 85–96 (2014)
36. Grzymala-Busse, J., Goodwin, L., Grzymala-Busse, W., Zheng, X.: Handling missing attribute values in preterm birth data sets. In: 10th International Conference of Rough Sets and Fuzzy Sets and Data Mining and Granular Computing(RSFDGrC05), pp. 342–351 (2005)

37. Grzymala-Busse, J.W., Hu, M.: A comparison of several approaches to missing attribute values in data mining. In: Ziarko, W., Yao, Y.Y. (eds.) Rough Sets and Current Trends in Computing, Lecture Notes in Computer Science, vol. 2005, pp. 378–385. Springer, Berlin (2000)
38. Howell, D.: The analysis of missing data. SAGE Publications Ltd, London (2007)
39. Hruschka Jr, E.R., Ebecken, N.F.F.: Missing values prediction with k2. Intell. Data Anal. **6**(6), 557–566 (2002)
40. Hulse, J.V., Khoshgoftaar, T.M.: Incomplete-case nearest neighbor imputation in software measurement data. Inf. Sci. **259**, 596–610 (2014)
41. Ingsrisawang, L., Potawee, D.: Multiple imputation for missing data in repeated measurements using MCMC and copulas, pp. 1606–1610 (2012)
42. Ishioka, T.: Imputation of missing values for unsupervised data using the proximity in random forests. In: eLmL 2013, The 5th International Conference on Mobile, Hybrid, and On-line Learning, pp. 30–36 (2013)
43. Jamshidian, M., Jalal, S., Jansen, C.: Missmech: an R package for testing homoscedasticity, multivariate normality, and missing completely at random (mcar). J. Stat. Softw. **56**(6), 1–31 (2014)
44. Joenssen, D.W., Bankhofer, U.: Hot deck methods for imputing missing data: the effects of limiting donor usage. In: Proceedings of the 8th International Conference on Machine Learning and Data Mining in Pattern Recognition, MLDM'12, pp. 63–75 (2012)
45. Juhola, M., Laurikkala, J.: Missing values: how many can they be to preserve classification reliability? Artif. Intell. Rev. **40**(3), 231–245 (2013)
46. Keerin, P., Kurutach, W., Boongoen, T.: Cluster-based knn missing value imputation for dna microarray data. In: Systems, Man, and Cybernetics (SMC), 2012 IEEE International Conference on, pp. 445–450. IEEE (2012)
47. Keerin, P., Kurutach, W., Boongoen, T.: An improvement of missing value imputation in dna microarray data using cluster-based lls method. In: Communications and Information Technologies (ISCIT), 2013 13th International Symposium on, pp. 559–564 (2013)
48. Khan, S.S., Hoey, J., Lizotte, D.J.: Bayesian multiple imputation approaches for one-class classification. In: Kosseim, L., Inkpen, D. (eds.) Advances in Artificial Intelligence - 25th Canadian Conference on Artificial Intelligence, Canadian AI 2012, Toronto, ON, Canada, Proceedings, pp. 331–336. 28–30 May 2012
49. Kim, H., Golub, G.H., Park, H.: Missing value estimation for dna microarray gene expression data: local least squares imputation. Bioinform. **21**(2), 187–198 (2005)
50. Krzanowski, W.: Multiple discriminant analysis in the presence of mixed continuous and categorical data. Comput. Math. Appl. **12**(2, Part A), 179–185 (1986)
51. Kwak, N., Choi, C.H.: Input feature selection by mutual information based on parzen window. IEEE Trans. Pattern Anal. Mach. Intell. **24**(12), 1667–1671 (2002)
52. Kwak, N., Choi, C.H.: Input feature selection for classification problems. IEEE Trans. Neural Networks **13**(1), 143–159 (2002)
53. Li, D., Deogun, J., Spaulding, W., Shuart, B.: Towards missing data imputation: a study of fuzzy k-means clustering method. In: 4th International Conference of Rough Sets and Current Trends in Computing (RSCTC04), pp. 573–579 (2004)
54. Little, R.J.A., Rubin, D.B.: Statistical Analysis with Missing Data, 1st edn. Wiley Series in Probability and Statistics, New York (1987)
55. Little, R.J.A., Schluchter, M.D.: Maximum likelihood estimation for mixed continuous and categorical data with missing values. Biometrika **72**, 497–512 (1985)
56. Lu, X., Si, J., Pan, L., Zhao, Y.: Imputation of missing data using ensemble algorithms. In: Fuzzy Systems and Knowledge Discovery (FSKD), 2011 8th International Conference on, vol. 2, pp. 1312–1315 (2011)
57. McLachlan, G.: Discriminant Analysis and Statistical Pattern Recognition. Wiley, New York(2004)
58. Merlin, P., Sorjamaa, A., Maillet, B., Lendasse, A.: X-SOM and L-SOM: a double classification approach for missing value imputation. Neurocomputing **73**(7–9), 1103–1108 (2010)

59. Michalksi, R., Mozetic, I., Lavrac, N.: The multipurpose incremental learning system AQ15 and its testing application to three medical domains. In: 5th INational Conference on Artificial Intelligence (AAAI86), pp. 1041–1045 (1986)
60. Miyakoshi, Y., Kato, S.: Missing value imputation method by using Bayesian network with weighted learning. IEEJ Trans. Electron. Inf. Syst. **132**, 299–305 (2012)
61. Moller, F.: A scaled conjugate gradient algorithm for fast supervised learning. Neural Networks **6**, 525–533 (1990)
62. Oba, S., aki Sato, M., Takemasa, I., Monden, M., ichi Matsubara, K., Ishii, S.: A bayesian missing value estimation method for gene expression profile data. Bioinform. **19**(16), 2088–2096 (2003)
63. Ouyang, M., Welsh, W.J., Georgopoulos, P.: Gaussian mixture clustering and imputation of microarray data. Bioinform. **20**(6), 917–923 (2004)
64. Panigrahi, L., Ranjan, R., Das, K., Mishra, D.: Removal and interpolation of missing values using wavelet neural network for heterogeneous data sets. In: Proceedings of the International Conference on Advances in Computing, Communications and Informatics, ICACCI '12, pp. 1004–1009 (2012)
65. Patil, B., Joshi, R., Toshniwal, D.: Missing value imputation based on k-mean clustering with weighted distance. In: Ranka, S., Banerjee, A., Biswas, K., Dua, S., Mishra, P., Moona, R., Poon, S.H., Wang, C.L. (eds.) Contemporary Computing, Communications in Computer and Information Science, vol. 94, pp. 600–609. Springer, Berlin (2010)
66. Peng, H., Long, F., Ding, C.: Feature selection based on mutual information: Criteria of max-dependency, max-relevance, and min-redundancy. IEEE Trans. Pattern Anal. Mach. Intell. 27(8), pp. 1226–1238 (2005)
67. Pham, D.T., Afify, A.A.: Rules-6: a simple rule induction algorithm for supporting decision making. In: Industrial Electronics Society, 2005. IECON 2005. 31st Annual Conference of IEEE, pp. 2184–2189 (2005)
68. Pham, D.T., Afify, A.A.: SRI: a scalable rule induction algorithm. Proc. Inst. Mech. Eng. [C]: J. Mech. Eng. Sci. **220**, 537–552 (2006)
69. Plat, J.: A resource allocating network for function interpolation. Neural Comput. **3**(2), 213–225 (1991)
70. Platt, J.C.: Fast training of support vector machines using sequential minimal optimization. In: Advances in Kernel Methods: Support Vector Learning, pp. 185–208. MIT Press, Cambridge (1999)
71. Pyle, D.: Data Preparation for Data Mining. Morgan Kaufmann Publishers Inc., San Francisco (1999)
72. Qin, Y., Zhang, S., Zhang, C.: Combining knn imputation and bootstrap calibrated empirical likelihood for incomplete data analysis. Int. J. Data Warehouse. Min. **6**(4), 61–73 (2010)
73. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, San Francisco (1993)
74. Rahman, G., Islam, Z.: A decision tree-based missing value imputation technique for data pre-processing. In: Proceedings of the 9th Australasian Data Mining Conference - Volume 121, AusDM '11, pp. 41–50 (2011)
75. Rahman, M., Islam, M.: KDMI: a novel method for missing values imputation using two levels of horizontal partitioning in a data set. In: Motoda, H., Wu, Z., Cao, L., Zaiane, O., Yao, M., Wang, W. (eds.) Advanced Data Mining and Applications. Lecture Notes in Computer Science, vol. 8347, pp. 250–263. Springer, Berlin (2013)
76. Rahman, M.G., Islam, M.Z.: Missing value imputation using decision trees and decision forests by splitting and merging records: two novel techniques. Know.-Based Syst. **53**, 51–65 (2013)
77. Rahman, M.G., Islam, M.Z.: Fimus: a framework for imputing missing values using co-appearance, correlation and similarity analysis. Know.-Based Syst. **56**, 311–327 (2014)
78. Royston, P., White, I.R.: Multiple imputation by chained equations (MICE): implementation in STATA. J. Stat. Softw. **45**(4), 1–20 (2011)
79. Rubin, D.B.: Inference and missing data. Biometrika **63**(3), 581–592 (1976)

80. Rubin, D.B.: Multiple Imputation for Nonresponse in Surveys. Wiley, New York (1987)
81. Safarinejadian, B., Menhaj, M., Karrari, M.: A distributed EM algorithm to estimate the parameters of a finite mixture of components. Knowl. Inf. Syst. **23**(3), 267–292 (2010)
82. Schafer, J.L.: Analysis of Incomplete Multivariate Data. Chapman & Hall, London (1997)
83. Schafer, J.L., Olsen, M.K.: Multiple imputation for multivariate missing-data problems: a data analyst's perspective. Multivar. Behav. Res. **33**(4), 545–571 (1998)
84. Scheuren, F.: Multiple imputation: how it began and continues. Am. Stat. **59**, 315–319 (2005)
85. Schneider, T.: Analysis of incomplete climate data: estimation of mean values and covariance matrices and imputation of missing values. J. Clim. **14**, 853–871 (2001)
86. Schomaker, M., Heumann, C.: Model selection and model averaging after multiple imputation. Comput. Stat. Data Anal. **71**, 758–770 (2014)
87. Sehgal, M.S.B., Gondal, I., Dooley, L.: Collateral missing value imputation: a new robust missing value estimation algorithm for microarray data. Bioinform. **21**(10), 2417–2423 (2005)
88. Silva-Ramírez, E.L., Pino-Mejías, R., López-Coello, M., Cubiles-de-la Vega, M.D.: Missing value imputation on missing completely at random data using multilayer perceptrons. Neural Networks **24**(1), 121–129 (2011)
89. Simński, K.: Rough fuzzy subspace clustering for data with missing values. Comput. Inform. **33**(1), 131–153 (2014)
90. Somasundaram, R., Nedunchezhian, R.: Radial basis function network dependent exclusive mutual interpolation for missing value imputation. J. Comput. Sci. **9**(3), 327–334 (2013)
91. Tanner, M.A., Wong, W.: The calculation of posterior distributions by data augmentation. J. Am. Stat. Assoc. **82**, 528–540 (1987)
92. Ting, J., Yu, B., Yu, D., Ma, S.: Missing data analyses: a hybrid multiple imputation algorithm using gray system theory and entropy based on clustering. Appl. Intell. **40**(2), 376–388 (2014)
93. Troyanskaya, O., Cantor, M., Sherlock, G., Brown, P., Hastie, T., Tibshirani, R., Botstein, D., Altman, R.B.: Missing value estimation methods for dna microarrays. Bioinform. **17**(6), 520–525 (2001)
94. Unnebrink, K., Windeler, J.: Intention-to-treat: methods for dealing with missing values in clinical trials of progressively deteriorating diseases. Stat. Med. **20**(24), 3931–3946 (2001)
95. Vellido, A.: Missing data imputation through GTM as a mixture of t-distributions. Neural Networks **19**(10), 1624–1635 (2006)
96. Wang, H., Wang, S.: Mining incomplete survey data through classification. Knowl. Inf. Syst. 24(2), 221–233 (2010)
97. Williams, D., Liao, X., Xue, Y., Carin, L., Krishnapuram, B.: On classification with incomplete data. IEEE Trans. Pattern Anal. Mach. Intell. 29(3), 427–436 (2007)
98. Wilson, D.: Asymptotic properties of nearest neighbor rules using edited data. IEEE Trans. Syst. Man Cybern. **2**(3), 408–421 (1972)
99. Wong, A.K.C., Chiu, D.K.Y.: Synthesizing statistical knowledge from incomplete mixed-mode data. IEEE Trans. Pattern Anal. Mach. Intell. **9**(6), 796–805 (1987)
100. Wu, X., Urpani, D.: Induction by attribute elimination. IEEE Trans. Knowl. Data Eng. **11**(5), 805–812 (1999)
101. Zhang, S.: Nearest neighbor selection for iteratively knn imputation. J. Syst. Softw. **85**(11), 2541–2552 (2012)
102. Zhang, S., Wu, X., Zhu, M.: Efficient missing data imputation for supervised learning. In: Cognitive Informatics (ICCI), 2010 9th IEEE International Conference on, pp. 672–679 (2010)
103. Zheng, Z., Webb, G.I.: Lazy learning of bayesian rules. Machine Learning **41**(1), 53–84 (2000)
104. Zhu, B., He, C., Liatsis, P.: A robust missing value imputation method for noisy data. Appl. Intell. **36**(1), 61–74 (2012)
105. Zhu, X., Zhang, S., Jin, Z., Zhang, Z., Xu, Z.: Missing value estimation for mixed-attribute data sets. IEEE Transactions on Knowl. Data Eng. **23**(1), 110–121 (2011)