# Towards Efficient Imputation by Nearest-Neighbors: A Clustering-Based Approach

Eduardo R. Hruschka[1], Estevam R. Hruschka Jr.[2], and Nelson F. F. Ebecken[3]

[1] Universidade Católica de Santos (UniSantos), Brasil
erh@unisantos.br
[2] Universidade Federal de São Carlos (UFSCAR), Brasil
estevamr@terra.com.br
[3] COPPE / Universidade Federal do Rio de Janeiro, Brasil
nelson@ntt.ufrj.br

**Abstract.** This paper proposes and evaluates a nearest-neighbor method to substitute missing values in ordinal/continuous datasets. In a nutshell, the K-Means clustering algorithm is applied in the complete dataset (without missing values) before the imputation process by nearest-neighbors takes place. Then, the achieved cluster centroids are employed as training instances for the nearest-neighbor method. The proposed method is more efficient than the *traditional* nearest-neighbor method, and simulations performed in three benchmark datasets also indicate that it provides suitable imputations, both in terms of prediction and classification tasks.

## 1 Introduction

Missing values are a critical problem for data mining methods, which are usually not able to cope with them in an automatic fashion (without data preparation). In general, there are many approaches to deal with the problem of missing values: i) Ignore the instances/attributes containing missing values; ii) Substitute the missing values by a constant; iii) Use the mean or the mode of the instances as a substitution value; and iv) Get the most suitable value to fill the missing ones. The first approach involves removing the instances and/or attributes with missing values. Doing so, the waste of data may be considerable and incomplete datasets can lead to biased statistical analyses. The second approach assumes that all missing values represent the same value, usually leading to considerable distortions. The substitution by the mean/mode value is a common practice and sometimes can even lead to reasonable results. However, it does not take into consideration the between-variable relationships, which are important to the process of missing values substitution. Therefore, the best approach involves filling the missing values with the most suitable ones.

The substitution of missing values, also called *imputation*, should not change important characteristics of the dataset. In this sense, it is necessary to define the important characteristics to be maintained. Data mining methods usually explore relationships between variables and, thus, it is critical to preserve them, as far as possible, when replacing missing values [1]. In other words, the imputation goal is to carefully substitute missing values, trying to avoid the imputation of bias in the dataset. When imputation is performed in a suitable way, higher quality data are produced, and the data mining outcomes can even be improved.

Several imputation methods have been proposed in the literature, and good references can be found in [2]. Some recent works suggest that K-Nearest Neighbors (KNN) methods [3] can be useful for imputation [4,5]. KNN can be defined as a *lazy memory-based* learning process [6] in which the training data is stored in memory and analyzed to find answers to specific queries. As far as the whole training dataset is stored in memory, the computational effort may be high when maintaining a global model to process all queries. To alleviate this problem, a local model can be used to fit the training data only in a region around the location of the query to be answered. This approach originates the usually called *local learning KNN methods*. In addition, indexing methods [3] can also be useful to minimize the computational cost of KNN methods.

In this work, we propose and evaluate an imputation method based on the K-Means clustering algorithm [7,8], which is employed to improve the computational efficiency of imputations based on nearest-neighbors. The paper is organized as follows. The next section describes how the K-Means can be incorporated into the *classical* KNN, originating our proposed imputation method. Besides, Section 2 also describes advantages and disadvantages of such approach, both in terms of efficacy and efficiency. Section 3 reports simulation results in prediction and classification tasks. Finally, Section 4 describes our conclusions and points out some future work.

## 2 Incorporating K-Means into the Nearest-Neighbor Method

In this section, we first review how to employ the classical KNN for imputation. Then, it is theoretically compared, in terms of efficacy (imputation quality) and efficiency (computational cost), with a clustering-based KNN method. Imputation methods based on KNN assume that missing values of an instance can be substituted by means of the corresponding attribute values of similar complete instances. More specifically, let us consider that each instance is described by a set of N continuous/ordinal attributes. Thus, each instance can be represented by a vector $\mathbf{x}=[x_1,x_2,...,x_N]$. The distance between two vectors (instances) $\mathbf{x}$ and $\mathbf{y}$ will be here called $d(\mathbf{x},\mathbf{y})$. Besides, let us suppose that the *i-th* attribute value ($1 \leq i \leq N$) of vector $\mathbf{u}$ is missing. Imputation methods based on nearest-neighbors compute distances $d(\mathbf{u},\mathbf{y})$, for all $\mathbf{y} \neq \mathbf{u}$, $\mathbf{y}$ representing a complete instance, and use these distances to compute the value to be imputed in $u_i$. The Euclidean metric is commonly used to compute distances between instances and, considering that the value of the *i-th* attribute is missing, it is given by:

$$d(\mathbf{u},\mathbf{y})_E = \sqrt{(u_1 - y_1)^2 + \ldots + (u_{i-1} - y_{i-1})^2 + (u_{i+1} - y_{i+1})^2 + \ldots + (u_N - y_N)^2} \quad . \qquad (1)$$

In equation (1), the i-*th* attribute is not considered, because it is missing in $\mathbf{u}$. After computing the distances $d(\mathbf{u},\mathbf{y})$ for all $\mathbf{y} \neq \mathbf{u}$, $\mathbf{y}$ representing a complete instance, one or more instances (the more similar ones, which are the neighbors of **u**) are employed to complete $u_i$. In addition, although equation (1) just considers one missing value (in the *i-th* attribute), one observes that it can be easily generalized for instances with more missing values.

The imputation process by the K-Nearest Neighbor (KNN) method is simple, but it has provided encouraging results [4,5,9]. In clustering problems, this approach is particularly interesting, because in general the clustering process is also based on distances between vectors. In these cases, the inductive bias of the clustering and imputation methods is equal. Besides, KNN methods can be easily adapted to datasets formed by discrete attributes. To do so, one can, for example, substitute the Euclidean distance function by the Simple Matching Approach. However, depending on the dimensionality of the dataset in hand, it can become computationally time consuming. In this sense, it is memory intensive since the entire training set is stored. Besides, classification/estimation is slow since the distance between input (query) instances and those in the training set must be computed. Thus, KNN typically performs better for lower dimensional problems [12].

In this paper, we focus on classification problems, describing a more efficient KNN algorithm for missing values imputation that can be employed in datasets formed by continuous/ordinal attributes. In a nutshell, the K-Means algorithm is first applied in the complete instances **y,** leading to a set of clusters, whose mean vectors (centroids) are, in turn, used as training instances for the *traditional* KNN. From now on, we will call our method KMI (K-Means Imputation), and we propose to apply it separately in the instances of each class (further details in Section 2.4). In summary, the main steps of KMI are:

```
Algorithm KMI

1 Employ the K-Means Algorithm in the instances without
missing values (complete dataset), obtaining $K_M$ cluster
centroids (mean vectors);

2 According to the $K_M$ obtained centroids, find the corre-
sponding nearest one for each instance with missing val-
ues - equation (1) considering y as the centroids and u
as the instance with missing values;

3 Complete each missing value with the corresponding at-
tribute value of the nearest centroid.
```

## 2.1 Brief Review of the K-Means Algorithm

Clustering algorithms that involve the calculation of the mean (centroid) of each cluster are often referred to as K-Means algorithms [7]. These algorithms partition a dataset of N instances into $K_M$ clusters, minimizing the sum of distances among instances and their corresponding cluster centroids. The value of $K_M$ (number of clusters) is usually specified by the user. The distances can be calculated by means of equation (1), but since K-Means is applied in the complete dataset, all attributes are considered. In this work, we employ the K-Means algorithm depicted in Fig.1, where the convergence criterion can be defined either as the maximum number of iterations (t) of steps

2 and 3 or as a function of the difference between centroids of two consecutive iterations.

```
1. Generate a random initial partition of instances into $K_M$ nonempty clusters;
2. Compute the cluster centroids (mean vectors) of the current partition;
3. Assign each instance to the cluster with the nearest centroid;
4. If the convergence criterion has been satisfied, then stop; else, go to step 2.
```

**Fig. 1.** Employed K-Means Algorithm

## 2.2 Efficacy of KNN and KMI

In order to visualize possible advantages and disadvantages of KNN and KMI, let us consider the *pedagogical* dataset (formed by 4 Gaussian distributions, each one composed by 10 instances) depicted in Fig. 2. Both KNN and KMI are designed to fulfill the missing value (represented by ?) of instance **u**=[$u_1$,$u_2$,?]. To do so, these methods take into consideration the available information, i.e., the values of attributes 1 and 2 depicted in Fig. 2, where {G1,…,G4} represent four *natural* groups of similar instances. These groups can be *viewed* as four different classes. It is clear that **u** is more similar to the instances of G1 (as indicated by the arrow) than to the instances of the other groups. Therefore, it is assumed that one or more instances of this group should be employed to impute $u_3$.

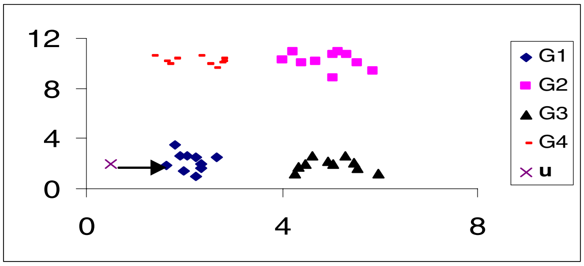

**Fig. 2.** Hypothetical dataset

Considering KNN, the most similar $K_N$ instances (neighbors) are employed to impute $u_3$. For instance, the mean vector of $K_N$ instances could be used to impute $u_3$. Let us call this mean vector as **m**=[$m_1$,$m_2$,$m_3$]. In this particular case, the third attribute value of the mean vector is imputed in $u_3$, i.e. $u_3$=$m_3$. In the hypothetical situation depicted in Fig. 2, a value of $K_N$ more than 10 may cause problems for KNN, because instances very dissimilar of **u** (belonging to other groups) would contribute to the imputation of $u_3$. This problem can be lessened by using the weighted imputation function expressed in Equation (2), where $\mathbf{y}^i$ are the $K_N$ most similar instances in relation to **u** and *j* is the attribute index of the missing value (*j=3* in our hypothetical case). Doing so, a vector **y** contributes to the imputation of $u_3$ according to its similarity to **u**. If d(**u**,$\mathbf{y}^i$)=0, then only instance $\mathbf{y}^i$ is employed to impute the missing value in

**u,** because in this case **u=y**$^i$. Equation (2) also lessens the problem caused by overestimated values of $K_N$ (number of neighbors in the traditional KNN). However, KNN computational cost continues high (this subject is addressed in Section 2.3).

$$u_j = \frac{\sum_{i=1}^{K_N} \frac{1}{d(\mathbf{u},\mathbf{y}^i)} y_j^i}{\sum_{i=1}^{K_N} \frac{1}{d(\mathbf{u},\mathbf{y}^i)}} \tag{2}$$

Let us now consider what happens with the application of KMI (K-Means + KNN). In principle, we will consider just the most similar cluster (according to its centroid) for the imputation process. In an ideal situation, the four groups depicted in Fig. 2 would be found by K-Means, and KMI would impute a suitable value in $u_3$, i.e. the corresponding value of the *mean vector* (centroid) of the most similar group (G1). In this case, if $K_N \leq 10$, KNN would perform a more accurate imputation - by Equation (2) – than KMI. However, for $K_N > 10$, imputation by KMI would be more accurate. Suppose now that $K_M \leq 4$ or that K-Means has not found the *natural* clusters {G1,…,G4}. In both cases, instances from different *natural* groups could be classified in the same cluster. For example, let us suppose that instances of G1 and G3 are *classified* in a single cluster. Under the perspective of imputing $u_3$, this single cluster would be interesting only if these two groups (G1 and G3) have similar values for $y_3$. To reduce this problem (without considering computational efficiency issues), one can overestimate (to a certain extent) the value of $K_M$ for two reasons: (i) it favors more compact clusters, formed by few instances, thus providing more accurate imputations; (ii) K-Means may *decrease* $K_M$, mainly in relation to particular centroid initializations. More specifically, K-means can find less than $K_M$ clusters when, in a set of $K_M$ centroids, at least one is farther from all instances than the others. In some situations, it can be interesting to employ more than one cluster for imputation. For example, let us consider the scenario in which two or more clusters have an equal distance from **u**. In this case, it would be interesting to consider all these clusters for imputation. Another alternative involves the application of equation (2) in the K-Means based imputation.

In theory, the challenge is to find suitable values for $K_N$ and $K_M$, because, in this case, both methods can provide good results. In practice, however, this is a hard task. Considering KNN, if $K_N$ is set too high, many dissimilar instances in relation to **u** are taken into consideration to impute its missing values. That could deteriorate the quality of the imputed value. On the other hand, if $K_N$ is set too low, the imputation could become badly biased, not appropriately reflecting sampling variability. Similarly, when KMI is concerned, if too high values of $K_M$ are chosen, small clusters are likely to be found, favoring imputations based on small subsets of instances. However, if $K_M$ is set too low, the number of instances in each cluster tends to be high, and very dissimilar instances in relation to **u** may be considered to impute its missing values. In summary, it is really difficult to find the optimal values for $K_M$ and $K_N$. Under this perspective, it is also important to evaluate the computational costs of KMI and KNN. This subject is addressed in the next section.

## 2.3 Computational Costs of KNN and KMI

In order to evaluate the computational costs of KNN and KMI, let us call: M as the amount of instances with at least one missing value; C as the amount of instances without missing values (complete dataset); $K_N$ as the number of neighbors for KNN; $K_M$ as the number of clusters for the K-Means Algorithm; and *t* as the number of iterations for K-Means.

Both methods require the computation of distances between vectors. These distances – given by equation (1) - represent the main computational cost of both KNN and KMI. The required comparisons do not represent significant computational costs when compared with the computation of distances. Thus, the estimated computational cost of KNN is $O(M \cdot C)$, whereas for KMI it is $O(t \cdot K_M \cdot C + M \cdot K_M)$. In practice, M and C are *a priori* known, and the values of both $K_M$ and *t* are usually defined by the user. With a little notation abuse, the estimated value of $K_M$ that equals the computational costs of KNN and KMI for a fixed *t* is:

$$K'_M = \frac{M \cdot C}{(t \cdot C + M)} \tag{3}$$

Equation (3) states that if $K_M > K'_M$ then KNN is computationally more efficient than KMI, whereas if $K_M < K'_M$ then KMI is more efficient. Similarly, it would be interesting to estimate $K'_N$. However, the number of neighbors ($K_N$) only affects the *comparison phase*, in which the nearest-neighbors are defined. Therefore, it does not significantly influences the computational cost of KNN and, in principle, $K'_N$ could not be estimated. One alternative to circumvent this problem involves comparing the computational costs of KNN and KMI in a situation in which both methods would provide the same efficacy. As previously described (Section 2.2), the efficacy of each method basically depends on the amount of instances employed for imputation. When KMI is concerned, in general it is interesting to consider only the most similar cluster in relation to **u** for imputation. Therefore, in order to provide a fair comparison between KNN and KMI, the value of $K_N$ should be equal to the number of instances of the most similar cluster. However, it is not easy to estimate this number, because it depends on the K-Means result. In this context, a reasonable approach involves assuming that K-Means would provide clusters formed by an equal number of instances, leading to the following relation:

$$K_M = \frac{C}{K_N} \tag{4}$$

In other words, equation (4) relates the number of clusters to the number of neighbors in a way such that both methods take into account approximately the same number of instances to impute the missing values. Now, similarly to the estimation of $K'_M$ - equation (3) - and using equation (4), it is possible to estimate $K'_N$:

$$K'_N = \frac{t \cdot C + M}{M} \tag{5}$$

Where $K'_N$ equals the computational cost of KNN and KMI, in such a way that both consider approximately an equal number of instances to perform the imputation of missing values. In summary, the computational costs of KNN and KMI depend on the proportion of instances with missing values in relation to the complete dataset, and on K-Means parameters.

### 2.4 Proposed Method

As described in Sections 2.2 and 2.3, KNN and KMI present advantages and disadvantages concerning both efficacy and efficiency, depending on the characteristics of the dataset in hand and on their parameter values ($K_M$, $t$, and $K_N$). In classification problems, we propose to separate the instances according to their corresponding classes before applying KMI. In this context, let us suppose a classification problem in which there are *X* classes. Thus, *X* pairs of {Missing,Complete} datasets {($M_1$,$C_1$), ($M_2$,$C_2$),…,($M_X$,$C_X$)} are employed for imputation. In this scenario, the hard task of choosing the value for $K_M$ is attenuated, because K-Means is employed in a *supervised* way, i.e., the information about the class is indirectly inserted in the imputation process. In other words, missing values are imputed only considering corresponding instances of the same class. In this sense, a reasonable approach is to set $K_M$=2, which is the minimum value for this parameter. Doing so, complete instances of each class are summarized by two cluster centroids (mean vectors), and the most similar centroid in relation to each instance with missing values is used for imputation.

Let us see what happens when the computational costs are concerned. To do so, it is not necessary to change the terminology defined in section 2.3. Instead, we assume that the imputation process is being performed in the instances of a single class *I*, and the concepts described in section 2.3 still remain valid for each pair ($M_I$,$C_I$). The only difference is that now we are evaluating KNN and KMI considering the instances of a single class. Anderberg [8] observes that, in general, a number of iterations $t \le 5$ usually will suffice to get suitable solutions by K-Means. Under this perspective, $\{t=5, K_M=2\} << \{C,M\}$ and, consequently, KMI becomes of *O(C+M)*, i.e. more efficient than KNN. Obviously, there is a trade-off between efficiency and efficacy (accuracy), and it is necessary to evaluate to what extent it is advantageous to employ more efficient algorithms like KMI. This assessment is highly dependent of the classification problem in hand. Thus, it is useful to perform empirical evaluation in some benchmark datasets. To do so, imputation results are evaluated both in terms of prediction and classification tasks.

## 3 Simulation Results

### 3.1 Theoretical Aspects

We are interested in evaluating our imputation method for classification problems in the context of data mining applications. In general, a dataset D is formed by instances with and without missing values. Remember that we call C (complete) the subset of instances of D that do not have missing values, and M (missing) the subset of instances of D with at least one missing value. In this context, imputation methods should carefully *complete* the missing values of M, originating a filled dataset F. In an ideal situation, the imputation method should fill the missing values, originating *filled values*, without inserting bias in the dataset. In a more realistic view, one tries to decrease the amount of inserted bias to acceptable levels, in a way that a dataset D'={C+F}, probably containing more information than D (in the sense that the attributes without missing values in M may contain important information), can be used for data mining (e.g. considering issues such as feature selection, combining multiple models, and so on).

There are two ways of measuring the *bias* inserted by an imputation method: in a prediction task and in a classification task. In a prediction task, one simulates missing values in C. Some known values are removed and then imputed. In this way, it is possible to evaluate how close the imputed values are to the real, known ones. The closer the imputed value to the real one, the better the imputation method is. This alternative is very efficient to compare different imputation methods, because it requires few computations after imputing values, but it does not allow estimating the classifier performance in D'. In other words, although this procedure is valid, the prediction results are not the only important issue to be analyzed. In this sense, the substitution process must also generate values that least distort the original characteristics of D, which are given by the between-variable relationships, defined by each particular classification algorithm. In a more practical view, the known values in M can contain important information, which, in turn, would be lost if their corresponding instances were discarded.

In practical data mining applications, one usually employs different classifiers, choosing the best one according to some criterion of model quality. In this work, we are interested in evaluating the influence of the proposed imputation method in relation to the Average Correct Classification Rate (ACCR) criterion. In fact, the assumption is that the best classifier (BC) - in relation to D and to the available classifiers – provides a suitable model for classifying instances of D. Therefore, it is also important to assess to what extent the imputed values adjust themselves to the BC model(s).

It is a common practice to evaluate classifier performance in a test set. The same concept can be adapted to evaluate the missing values substitution, considering C as the training set and F as the test set. In this context, one can measure the *inserted bias* by the following procedure:

1) Evaluate the classifier in a cross-validation process, using dataset C;
2) Evaluate the classifier in dataset F (test set) considering that C is the training set;
3) The estimated inserted bias is the difference between the results achieved in 2) and 1).

**Fig. 3.** Evaluating the imputation method in a classification context

Looking at Fig. 3, a positive bias is achieved when the Average Correct Classification Rate (ACCR) in F (step 2) is higher than in the cross validation process in C (step 1), i.e. the imputed values are likely to improve the classifier ACCR in D'. By the same token, a negative bias is inserted when the imputed values are likely to worsen the classifier ACCR. Finally, no bias is likely inserted when the accuracies in F and in the cross-validation process are equal (*ideal* situation).

One could argue that a cross-validation process should be performed in both datasets (C and F), comparing the corresponding results. However, it is possible to get different models in C and F. For instance, different decision trees (in terms of selected attributes), but with similar accuracies in a cross-validation process, may be obtained in C and F. In this case, a similar ACCR may not be achieved in D'. In addition, we are interested in evaluating if the imputation process preserves the between-variable relationships, which are defined by each particular classification model. In this sense, the complete dataset is more trustable and, consequently, is its associated model. In other words, verifying how the data in F adjust themselves to the model obtained in C allows estimating the classifier ACCR in D'. Besides, performing cross-validation in both datasets (C and F) is computationally more expensive, because it is necessary to get and evaluate each classifier several times for both C and F.

## 3.2 Methodology

The KMI was evaluated both in prediction and classification tasks, comparing the obtained results with the traditional KNN for imputation. In order to verify the efficacy of the proposed method (KMI), we compared it to KNN with weighted imputations – equation (2) – and assuming that $K_N=C/K_M$ provides fair comparisons – equation (4). Both methods were evaluated in three scenarios, simulating missing values in proportions of 30%, 50% and 70%, i.e. M=30%C, M=50%C and M=70%C. To do so, we simulated missing values in complete datasets, eliminating some values that are a priori known. In this sense, for each class some instances were randomly chosen, and one of their attribute values was randomly eliminated. Thus, the proportion of instances of each class is maintained in the employed datasets. After inserting missing values, the known values of each attribute were normalized into the range [0,1] to give attributes an equal weight. We performed simulations in 3 datasets that are benchmarks for data mining methods [10] (Iris Plants, Wisconsin Breast Cancer, and Pima Indians Diabetes) and Table 1 summarizes their main characteristics.

**Table 1.** Summary of Dataset Characteristics

| Dataset (# classes) | # instances/class | # attributes | Attribute Type |
|---|---|---|---|
| Iris Plants (3) | {50,50,50} | 4 | Continuous |
| Wisconsin Cancer (2) | {444,239} | 9 | Ordinal |
| Pima Diabetes (2) | {500,268} | 8 | Ordinal/Continuous |

The simulation results in the prediction task are shown in Table 2, where one observes that KNN has provided slightly better results (smaller average absolute differences between original and imputed values) in most of the performed simulations. As it can be seen in the sequel, these differences have not significantly influenced classification results, which is the most important aspect to be analyzed.

**Table 2.** Prediction Results: average values for abs(original-imputed)

| Metho d | 30% Iris | 50% Iris | 70% Iris | 30% Wisc. | 50% Wisc. | 70% Wisc. | 30% Pima | 50% Pima | 70% Pima |
|---|---|---|---|---|---|---|---|---|---|
| KNN | 0.22 | 0.19 | 0.26 | 1.15 | 1.12 | 1.05 | 12.65 | 14.85 | 17.38 |
| KMI | 0.21 | 0.21 | 0.28 | 1.22 | 1.14 | 1.11 | 12.73 | 15.52 | 17.33 |

The imputation process must also generate values that least distort the original characteristics of D, which are given by the between-variable relationships, defined by each particular classifier. To evaluate this aspect, we employ the methodology described in Section 3.1, using five classifiers: One Rule, Naïve Bayes, J4.8 Decision Tree, PART and Multilayer Perceptrons. These classifiers are popular in the data mining community, and make part of the WEKA System [11], which was used to perform our simulations, using its default parameters. In a nutshell, One Rule (1R) is a very efficient and simple method that often produces good rules for characterizing the structure in the data. It generates a one-level decision tree, which is expressed in the form of a set of rules that test just one selected attribute. Naive Bayes (NB) uses all attributes and allows them to make contributions to the decision that are equally important, and independent of one another given the class, leading to a simple scheme that works well in practice. J4.8 is the Weka´s implementation of an improved version of the popular C4.5 decision tree learner. PART is a method that provides rules from pruned partial decision trees built using C4.5. It combines the divide-and-conquer strategy of decision trees learning with the separate-and-conquer strategy for rule learning. In essence, to make a single rule, a pruned decision tree is built for the current set of instances. Then, the leaf with the largest coverage is made into a rule and the tree is discarded. Finally, Multilayer Perceptrons are feedforward neural networks (NN) that learn by means of backpropagation algorithms.

In order to estimate the classifier ACCR (Average Correct Classification Rate) in the complete datasets (C), we performed a ten-fold-cross validation (CV) process. Figures 4, 5, and 6 show the simulation results in each dataset, considering the employed classifiers. In these figures, 30%, 50% and 70% represent the proportion of missing values in each simulation, whereas F stand for the datasets filled by either KNN or KMI.

Figures 4, 5, and 6 show that KNN and KMI have provided good and similar classification results in the filled datasets. Since KMI is a more efficient algorithm, our most important results concern about the efficacy of KMI, which is similar to the one provided by KNN. Indeed, KNN has provided better classification results just in 40% of our simulations. Indeed, the most important differences, in terms of classification results in the filled datasets, were equal to 2.22%, 0.63%, and 3.90% in Iris, Wisconsin and Pima respectively. Considering the inserted bias (Fig. 3) both methods (KMI and KNN) have provided equal results in 44.44% of our simulations, whereas KMI has provided better results in 31.11% of our simulations, indicating that KMI provides imputations as suitable as KNN, but in a more efficient way.

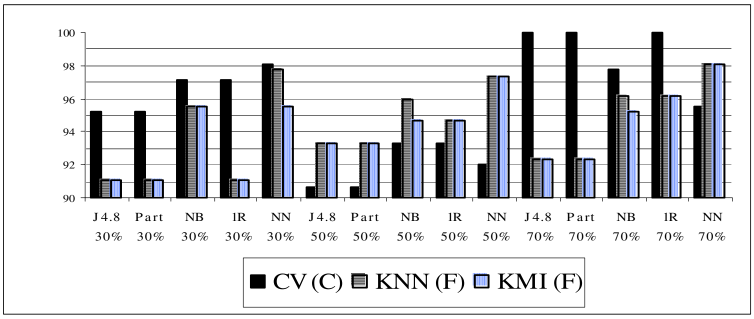

**Fig. 4.** Average Correct Classification Rate (ACCRs) - Iris Plants

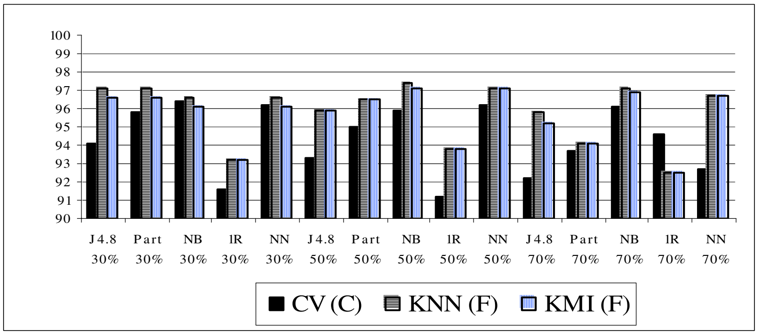

**Fig. 5.** Average Correct Classification Rate (ACCRs) - Wisconsin Breast Cancer

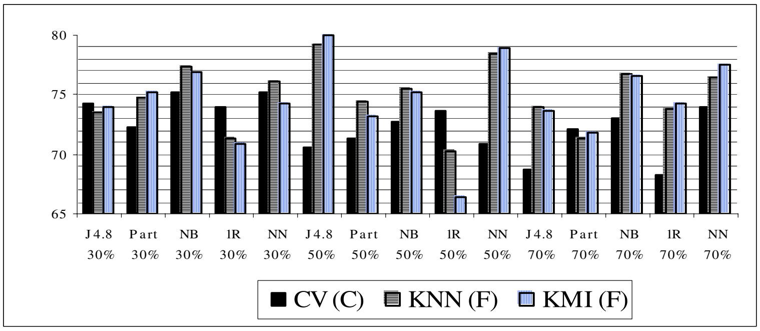

**Fig. 6.** Average Correct Classification Rates (ACCRs) - Pima Indians Diabetes

Considering the most important estimated inserted biases, Figures 4, 5, and 6 also show that KNN and KMI provided similar performances. Table 3 details the most important inserted biases (positive and negative ones), indicating that KMI has shown worse results just in Pima.

**Table 3.** Most Important Estimated Inserted Bias in Each Dataset: Bias (bold); imputation method(s) (KNN/KMI/both); classifier; proportion of missing values

| **Bias (%)** | **Iris Plants** (Fig. 4) | **Wisconsin** (Fig. 5) | **Pima** (Fig. 6) |
|---|---|---|---|
| Positive | **5.33**; both; NN; 50%. | **3.97**; both; NN; 70%. | **9.38**; KMI; J4.8; 50% |
| Negative | **7.62**; both; J4.8/PART; 70%. | **2.16**; both; 1R; 70%. | **7.29**; KMI; 1R; 50% |

## 4 Conclusions

This paper proposed modifications towards improving the efficiency of imputation methods based on nearest-neighbors. In this context, the K-Means algorithm is applied in the complete dataset (without missing values) before the imputation process by nearest-neighbors takes place. Subsequently, the cluster centroids obtained by K-Means are used as *training* instances for the nearest-neighbor method. Particularly, we focused on classification problems, for which we proposed to employ K-Means in a *supervised* way, i.e., missing values are imputed considering just corresponding instances of the same class. Performed simulations in three benchmark datasets indicate that the proposed method provides imputations as suitable as those obtained by means of the traditional K-Nearest Neighbor (KNN) method, both in terms of prediction and classification tasks. Thus, the proposed method is promising.

Our future work will concentrate on performing simulations on real-world datasets, comparing the obtained results with other leading imputation methods. We are also going to evaluate the application of the weighted imputation function in the instances of the clusters found by K-Means. Although it implies in an additional computational cost, the imputations are likely to be even more accurate. In addition, the computational cost can be further minimized by using other reduction techniques such as those proposed in [13]. Finally, we are going to investigate the substitution of K-Means by other efficient algorithms that automatically define the *optimal* number of clusters.

## Acknowledgments

Eduardo R. Hruschka acknowledges CNPq (proc. 301.353/03-4) for its financial support. Nelson F.F. Ebecken acknowledges both CNPq and FAPERJ for their financial support.

## References

1. Pyle, D., Data Preparation for Data Mining. Academic Press, 1999.
2. Little, R. & Rubin, D. B., Statistical Analysis with Missing Data. Wiley, New York, 1987.
3. Mitchell, T. M. Machine Learning. The McGraw-Hill Companies, Inc, 1997.
4. Hruschka, E. R., Hruschka Junior, E. R., Ebecken, N. F. F. Evaluating a Nearest-Neighbor Method to Substitute Continuous Missing Values. In: The 16th Australian Joint Conference on Artificial Intelligence, Lecture Notes in Artificial Intelligence, v. 2903, pp. 723-734, Springer, 2003.
5. Batista, G. E. A. P. & Monard, M. C., An Analysis of Four Missing Data Treatment Methods for Supervised Learning. Applied Artificial Intelligence. v.17, n.5-6, 519-534, 2003.
6. Atkeson, C. G., Moore, A. W., & Schaal, S., Locally Weighted Learning, Artificial Intelligence Review, 11:11-73, 1997.
7. Everitt, B.S., Landau, S., Leese, M., Cluster Analysis, Arnold Publishers, London, 2001.
8. Anderberg, M. R., Cluster Analysis for Applications, USA, Academic Press, Inc., 1973.
9. Troyanskaya, O. et al., Missing Value Estimation Methods for DNA Microarrays, Bioinformatics, v.17, no. 6, pp. 520-525, 2001.
10. Merz, C.J., Murphy, P.M., UCI Repository of Machine Learning Databases, http://www.ics.uci.edu, Irvine, CA, University of California, Department of Information and Computer Science.
11. Witten, I. H., Frank, E., Data Mining – Practical Machine Learning Tools and Techniques with Java Implementations, Morgan Kaufmann Publishers, USA, 2000.
12. Kennedy, R.L., Lee, Y., Roy, B.V., Reed, C.D., Lippmann,R. P., Solving Data Mining Problems through Pattern Recognition, Prentice Hall PTR, 1997.
13. Wilson, D.R., Martinez, T.R., Reduction Techniques for Instance-Based Learning Algorithms, Machine Learning, 38-3, pp. 257-286, Kluwer Academic Publishers, 2000.