The handle http://hdl.handle.net/1887/65632 holds various files of this Leiden University dissertation.

**Author:** Stein, B. van  
**Title:** Data driven modeling & optimization of industrial processes  
**Issue Date:** 2018-09-20
In the first steps of the data driven framework, data-preprocessing is of high importance. Many issues can occur in real-world applications such as erroneous data, missing data, noise, cumbersome code, alignment issues between different devices, encodings and many more. Here we focus on the problem of missing values. Missing values play an important role in the data-preprocessing step as they negatively influence the usability of the data and the results of many data driven algorithms. Dealing correctly with these missing or even erroneous values is not a trivial task. Naively removing all records with missing values will lead to bias in the data and a loss of information, while repairing or replacing the missing values might also result in bias, erroneous values and mis-information. In this chapter several techniques of visualizing missing values are proposed to gain a better understanding of the missing value mechanisms. In addition an algorithm is proposed to effectively repair missing values in an iterative column-wise fashion.

In this chapter the challenges of missing values are examined and for both the exploration and imputation of these missing values, a solution is proposed. The content of this chapter is primarily based on the two publications [13] and [14].
3. MISSING VALUE ANALYSIS AND IMPUTATION

3.1 Introduction

In industrial processes and many other real-world applications, data points are collected to gain insight into the process and to make important decisions. Understanding and making predictions for these processes are vital for their optimization. Missing values in the collected data cause additional problems in building predictive models and applying them to fresh data. Unfortunately, missing values are very common and occur in many processes, for example, sensors that collect data from a production line may fail; a physician that examines a patient might skip some tests; questionnaires used in market surveys often contain unanswered questions, etc. This problem leads to the following questions:

1. What are the causes for missing values and can patterns of missing values be observed?

2. How to build high quality models for classification and regression, when some values in the training set are missing?

Gaining more insights into the patterns of the missing values is an important factor for selecting algorithms that are appropriate for a given data set. A theory about missing value patterns and mechanisms already exists, but the existing theory is insufficient to gain a clear understanding of each possible pattern of missing values because it only defines a small set of possibilities.

Proposed is an extension to the current theory, covering all patterns of missing values occurring in a data set, ranging from a completely Univariate Pattern to a completely Arbitrary Pattern. Using this new concept, a greedy algorithm that analyzes data sets is proposed, together with various visualization techniques that provide a clear overview of the patterns of missing values occurring in the data set. Besides analyzing missing values, it is also interesting that the same techniques can be used to analyze the patterns of occurrence of a specific value. For example, the patterns in sparse data sets (where 0 would be the unique value to analyze, or even more interesting where all non-zero values are analyzed).

After analyzing the missing value patterns, these missing values need to be dealt with in such a way that predictive models can be fitted on the data set. There
are several methods developed for tackling this imputation problem, see e.g., \[20, 21, 16, 22, 23\]. The most common method, *imputation*, reconstructs the missing values with help of various estimates such as means, medians, or simple regression models which predict the missing values. Proposed is a more sophisticated approach, *Incremental Attribute Regression Imputation* (IARI) which prioritizes all attributes with missing values and then iteratively “repairs” each of them, one by one, using values of all attributes that have no missing values or are already repaired, as predictors. Additionally, the target variable is also used as a predictor in the repair process. Repairing an attribute is achieved by constructing a regression model and applying it for estimation of missing values. The Random Forest algorithm, \[24, 18\], is used in these experiments due to its accuracy, robustness, and versatility: it can be used to model both numerical and categorical variables. Obviously, after repairing all attributes with missing values a final model for the original target variable can be trained on the repaired training set.

The proposed algorithm is evaluated using five well-known data sets: *Digits*, *Page Blocks*, *Concrete*, and *CoverType* from the UCI Machine Learning Repository, \[25\], and *Housing 16H* from mldata.org \[26\], first removing some values at random, and then reconstructing them with help of IARI and several common imputation algorithms. Finally the quality of these imputation methods is compared by measuring the accuracy of regression and classification models trained on the reconstructed data sets. The results demonstrate that in most cases, no matter how many attributes were spoiled and how severely, the IARI algorithm outperformed other imputation methods both in terms of the accuracy of the final models and the accuracy of imputation. On the other hand, the IARI algorithm is computationally very demanding—it builds as many Random Forests as the number of attributes that should be repaired. Fortunately, due to the parallel nature of the Random Forest algorithm, the runtime of the IARI algorithm can be easily reduced by running it on a system with multiple cores or CPUs.

This chapter first elaborates on possible patterns and causes of missing values in data sets and how to analyze and visualize these patterns. Later on in the chapter the IARI algorithm is explained in detail and results from various experiments are presented and discussed.
3. MISSING VALUE ANALYSIS AND IMPUTATION

3.2 Missing Value Analysis

In the following two subsections an overview of common definitions of several mechanisms behind missing data is given, and several new concepts of “patterns of missingness” are introduced.

3.2.1 Missing Data Types

Rubin et al [15] defined three major classes of missing values: Missing Completely At Random (MCAR), Missing At Random (MAR), and Missing Not At Random (MNAR). Informally, we say that values in a data set are Missing Completely At Random (MCAR), if the probability distribution of “being missing” is completely independent on the observed or missing data. When the probability distribution of “being missing” somehow depends on the observed (non-missing) values, then we talk about the Missing At Random (MAR) scenario. Finally, when “being missing” depends on the actual, unobserved values, then we talk about the Missing Not At Random (MNAR) scenario.

To illustrate these three definitions, let us consider data of patients collected in a hospital. When a doctor decides not to measure patient’s body temperature because she can already see that the temperature is too high, then we have the MNAR scenario - the decision of not measuring the parameter depends on its actual value. On the other hand, if the temperature is systematically measured, but from time to time the data registration process malfunctions (independently on the measured values), then we have the MCAR scenario. Finally, if the doctor has a habit of not measuring the temperature of patients with high blood pressure (and blood pressure is always registered), then we have a MAR scenario.

Formally, the three scenarios can be summarized in the following definition. [18]:

Definition 3.1 Let $y$ denote a target attribute, $X$ a matrix of input attributes with missing values, $X_{obs}$ the observed entries in $X$, $Z = (y, X)$, $Z_{obs} = (y, X_{obs})$. Additionally, let $R$ denote an indicator matrix with $ij$th entry 1 if $x_{ij}$ is missing and 0 otherwise.
We say that the data is **Missing Completely At Random** if:

\[
Pr(R|Z, \theta) = Pr(R|\theta).
\]

We say that the data is **Missing At Random** if:

\[
Pr(R|Z, \theta) = Pr(R|Z_{obs}, \theta).
\]

We say that the data is **Missing Not At Random** if it is not Missing at Random. (Here we assume that probability distributions are parametrized by parameters \(\theta\).)

### 3.2.2 Patterns of Missing Values

In addition to some general probabilistic mechanisms behind missing values, one can also look at the shape of missing values in the data table. In the literature \[^{[16]}\], three definitions of missing value patterns exist; namely **Univariate**, **Monotone** and **Arbitrary** pattern.

A **univariate** pattern (Figure 3.1a) of missing values means that one or several attributes (columns) contain missing values in exactly the same records and no other values are missing. When attributes can be organized in several groups \(G_1, \ldots, G_k\), such that each group forms a univariate pattern and records with missing values in \(G_i\) have also missing values in \(G_{i−1}\), for \(i = k, \ldots, 2\), then we have a **monotone** pattern (Figure 3.1b). An **arbitrary** pattern, is anything else. Obviously, in order to visualize patterns of missing values, one has to permute columns and rows of the data matrix to create “rectangular regions of missingness”.

It is very important to understand the patterns of missing values in a data set, because they might provide insight into why values are missing and relations of attributes that are missing in groups. As an example: A camera system fails to recognize the bar-code of a certain product, which in turn makes it impossible for the next two sensors to save their measurements to the database, resulting in two missing values. Additionally, identifying important patterns of missing values in the data can lead to using a different strategy for handling these missing values. However, in reality there are many more patterns that are now falling
under the category *arbitrary pattern*, but that are not arbitrary at all. Consider a data set with a Monotone pattern of missing values, now remove one value from a column that does not contain any missing values yet. The data set with the extra removed value falls under the arbitrary category, while in reality the data set is almost completely falling into the category of a monotone pattern. Another example, imagine that a survey takes place led by two volunteers, both volunteers ask the same ten questions to a hundred different people, but volunteer $a$ asks the questions in order, and volunteer $b$ asks the questions in reverse order. Due to time limitations, people start to drop out after the sixth question. The combined data set seems to have an arbitrary pattern of missing values, while if we look more closely we can identify two partitions of the data with both one monotone pattern of missing values.

To fill the gap between the definitions of missing value patterns, we introduce the concept of *Mixtures of Monotone patterns*. This requires a more precise definition of the *Univariate* and *Monotone* patterns.

Let us consider a data set $D$ of size $N \times k$, with $N$ the number of records in $D$ and $k$ the number of attributes in $D$, and a missing indicator matrix $I$. Here, $I_{ij} = 1$ if $D_{ij}$ is missing, and 0 otherwise.

**Definition 3.2 (Univariate pattern)** *Missing values in $D$ form a univariate pattern if and only if there exists a set of attributes $A$ such that:*

$$\forall x \in D : \{a : x_a \text{ is missing}\} = A \text{ or } \emptyset$$

*So for all records in $D$, the record has either missing values in all attributes in the attribute set $A$ or the record has no missing values.*

**Definition 3.3 (Monotone pattern)** *A data set $D$ has missing values in a monotone pattern if and only if there exists an ordering of all attributes $A, a_1 \ldots a_k$ such that:*

$$\forall i \in \{1, \ldots, N\}, \forall j \in \{1, \ldots, k\} : I_{i,j} = 0 \Rightarrow I_{i,j+1} = 0, \ldots, I_{i,k} = 0$$

Note that Definition 3.3 is a generalization of Definition 3.2, in other words, a
3.2 Missing Value Analysis

(a) Missing values in a univariate pattern.

(b) Missing values in a monotone pattern.

(c) Missing values in a k-monotone mixture pattern.

Figure 3.1: Data set of records (y-axis) and attributes (x-axis) with missing values denoted by the colored bars.

A univariate pattern is a special case of a monotone pattern. A monotone pattern can also be seen as a collection of record groups, where each group of records has a univariate pattern of missing values. For example, a data set has twenty attributes and forty records, five of the records have attribute one and five missing this is denoted as: \((1, 5)\). Ten records have attributes one, five and nineteen missing: \((1, 5, 19)\), and twenty records have only attribute five missing: \((5)\). The remaining five records are complete. The complete data set has a monotone pattern of missing values, which can be denotes as set \(P = \{(5), (1, 5), (1, 5, 19)\}\). Each element in \(P\) stands for a univariate pattern that holds within a subset of the complete data set.

This way we can further generalize into a Mixture of Monotone patterns (Figure 3.1c).

**Definition 3.4 (k-Monotone Mixture Pattern)** A data set \(D\) has missing values in a k-monotone mixture pattern if and only if here is a partitioning of \(D\), \(S = S_0 \ldots S_{k-1}\) of size \(k\) such that \(S_0 \cup S_1 \cdots \cup S_{k-1} = D\) and \(\forall S_i, S_j \in S, i \neq j : S_i \cap S_j = \emptyset\) and \(\forall S_i \in S : S_i\) has values missing in a monotone pattern.

A univariate pattern can be seen as a rectangular area of missingness, a monotone pattern can be seen as a stack of adjacent rectangular regions and a monotone mixture pattern can be seen as a union of disjoint monotone patterns.
3. MISSING VALUE ANALYSIS AND IMPUTATION

Note that any data set has values missing in a \( k \)-Monotone Mixture Pattern where \( k \leq N \). When there exists a 1-Monotone Mixture Pattern, the pattern is completely monotone, when there exists only a high \( k \) mixture pattern, the pattern is close to arbitrary. In this manner, a transition between completely monotone and arbitrary patterns can be identified.

3.2.3 Analyzing Missing Value Patterns

It is possible to analyze a data set and identify the existing monotone mixture patterns using our novel MMP-Finder (Algorithm 3.1). In this algorithm, first a dictionary with all existing monotone patterns of missing values is built and sorted by the number of rows per pattern. Then, mixtures of monotone patterns are constructed by adding the next monotone pattern to an already existing mixture, or by defining a new mixture. The MMP-Finder uses a greedy approach to construct the mixtures of monotone patterns.

The complexity of the proposed greedy approach is \( O(n + m^2) \), where \( n \) is the number of records and \( m \) is the number of unique sets of missing attributes. Of course \( m \leq n \), since every record can have a unique set of attributes missing, but usually \( m \) is much smaller than \( n \).

Using the MMP-Finder, all identified monotone patterns in the partitions of data set \( X \) are returned, together with the number of records and record indexes that belong to each monotone pattern. Notice that the solution returned is not a unique solution, it is possible that a specific univariate pattern can belong to multiple monotone patterns. For example, two monotone pattern sets are defined: \{ (1,5), (1,5,8) \} and \{ (4,5), (4,5,9) \}, the next univariate pattern that occurs is (5), this pattern might belong to the first monotone pattern set, or the second. The proposed algorithm handles these choices in a greedy manner, the univariate patterns are handled in an order depending on the coverage, the pattern that covers most records is handled first, the pattern that covers the least records is handled last. This way it is very likely to identify “the biggest” monotone pattern. Since the missingness mechanism is usually not known, it is impossible to find the “correct” monotone patterns.
Algorithm 3.1 MMP-Finder

**Given:** A training set $X$ with input attributes $x_0, \ldots, x_n$ containing missing values, a target attribute $y$

{Create a dictionary $CM$ with all unique missing attribute combinations}

$CM = \text{unique}(\text{foreach records } x_i \in X: \text{Attr}_\text{missing}(x_i))$

{For each combination store the records with that combination of missing attributes}

$RecordsPerCombination = X[\text{Attr}_\text{missing}(CM)]$

{Sort the combinations by size}

$sortedComb = \text{sort}(CM)$

$Mixtures = []; MixtureRecords = []$

{Construct the mixtures}

for all $comb \in sortedComb$ do

for all $M \in Mixtures$ do

{If the comb. is a sub or super set for all combinations in $M$ add it to $M$}

if $\forall c \in M : comb \subseteq c \lor comb \supseteq c$ then

$Mixtures[M].append(comb)$

$MixtureRecords_M.append(RecordsPerCombination[comb])$

$Added = True$

break

end if

end for

if $\neg Added$ then

{Add a new Mixture}

$Mixtures.append([comb])$

$MixtureRecords.append([RecordsPerCombination[comb]])$

end if

end for

return $Mixtures$
3. MISSING VALUE ANALYSIS AND IMPUTATION

Table 3.1: Textual summaries for data sets with Missing Values

<table>
<thead>
<tr>
<th>Data set</th>
<th>#Mixtures</th>
<th>Ratio of each mixture</th>
<th>Miss.%</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Post-oper.</td>
<td>1</td>
<td>[ 1.0 ]</td>
<td>0.033</td>
<td>Monotone</td>
</tr>
<tr>
<td>Wisconsin</td>
<td>1</td>
<td>[ 1.0 ]</td>
<td>0.023</td>
<td>Monotone</td>
</tr>
<tr>
<td>Dermato.</td>
<td>1</td>
<td>[ 1.0 ]</td>
<td>0.022</td>
<td>Monotone</td>
</tr>
<tr>
<td>Cleveland</td>
<td>2</td>
<td>[ 0.667 0.33 ]</td>
<td>0.020</td>
<td>Two monotone patterns</td>
</tr>
<tr>
<td>Adult</td>
<td>2</td>
<td>[ 0.776 0.224 ]</td>
<td>0.074</td>
<td>Two monotone patterns</td>
</tr>
<tr>
<td>Census</td>
<td>7</td>
<td>[ 0.948 0.030 0.006 0.001 ...]</td>
<td>0.527</td>
<td>Mostly monotone</td>
</tr>
<tr>
<td>Automobile</td>
<td>4</td>
<td>[ 0.826 0.087 0.043 0.043 ]</td>
<td>0.224</td>
<td>Mostly monotone</td>
</tr>
<tr>
<td>Hepatitis</td>
<td>7</td>
<td>[ 0.707 0.093 0.093 0.053 ...]</td>
<td>0.484</td>
<td>70% mono., 30% rand.</td>
</tr>
<tr>
<td>Mammogr.</td>
<td>5</td>
<td>[ 0.550 0.244 0.160 0.038 ...]</td>
<td>0.136</td>
<td>Monotone mixture</td>
</tr>
<tr>
<td>Bands</td>
<td>18</td>
<td>[ 0.39 0.259 0.086 0.086 ...]</td>
<td>0.323</td>
<td>60% two patterns</td>
</tr>
<tr>
<td>Wiki</td>
<td>116</td>
<td>[ 0.503 0.091 0.030 0.016 ...]</td>
<td>0.807</td>
<td>50% mono., 50% rand.</td>
</tr>
<tr>
<td>Marketing</td>
<td>39</td>
<td>[ 0.353 0.128 0.112 0.111 ...]</td>
<td>0.235</td>
<td>Random</td>
</tr>
<tr>
<td>Horse-colic</td>
<td>82</td>
<td>[ 0.221 0.061 0.058 0.044 ...]</td>
<td>0.981</td>
<td>Random</td>
</tr>
</tbody>
</table>

Once the monotone patterns and their support are known, it is easier to verify why certain attributes contain missing values, and whether there are relations between the various attributes inside the monotone patterns. This can not only provide valuable insight, but also help in choosing a good imputation or modeling algorithm.

3.2.4 Analysis of Existing Data Sets

Fourteen data sets with missing values from the UCI machine learning repository [25] were analyzed using Algorithm 3.1. The output of the algorithm is shown in Table 3.1 and a visualization of the result is provided in Figure 3.2 and Figure 3.3. The visualization and textual summaries are generated directly from the output of the MMP-finder algorithm.

In Figure 3.2 the monotone mixture patterns found in the Wiki data set can be observed using two kinds of visualization techniques. In Figure 3.2a each record in the data set that contains missing values is labeled with a color and a position on the x axis. This way it is easy to observe where several monotone mixture patterns are located in the data set and if there are specific regions in the data set where these patterns occur. Additionally, the horizontal length of each bar depends on the number of attributes that are missing. For each mixture, the longest
3.2 Missing Value Analysis

(a) Visualization of missing values per record. Each column (color) represents a mixture of monotone patterns, the length of each bar is proportionate to the number of missing attributes versus the maximum number of missing attributes in its mixture.

(b) Visualization by the number of records affected per pattern. Each color is a mixture of monotone patterns, each bar is a monotone pattern.

Figure 3.2: First five Monotone Mixture Patterns for the Wiki data set.

This visualization technique, presenting the various monotone patterns in a data set, can be useful in understanding the underlying missing data mechanisms. For example, in the Wiki data set, the two monotone patterns that cover most of the records are located in a very specific order in the data set, which might be relevant information regarding the missing data mechanism. Even more specific, the three most occurring univariate patterns occur exactly after each other in the data set. In Figure 3.2b, the same patterns can be observed, but now in a histogram plot. The distribution of records belonging to each univariate pattern can be observed and the largest monotone patterns in terms of the number of records and in terms of the number of univariate patterns can be identified easily. Using this visualization technique it is easy to observe the different distributions in between the patterns. In Figure 3.3, a visualization of all the data sets with natural missing values is shown using the first visualization technique.
3. MISSING VALUE ANALYSIS AND IMPUTATION

![Visualization of data sets with natural occurring missing values.](image)

Figure 3.3: Visualization of data sets with natural occurring missing values.

### 3.3 Incremental Attribute Regression Imputation

Now that the patterns of missing values can be analyzed, the next step is repairing these missing values. In this section the proposed IARI algorithm is explained in detail and experimental results are discussed.

There are two ideas behind our method for incremental repair of training sets. First, attributes with missing values are repaired one by one, according to the priority of the attribute. The attribute with the highest priority is repaired first, the attribute with the lowest priority is repaired last. Second, the data used for repairing an attribute include all attributes that are already repaired and addi-
3.3 Incremental Attribute Regression Imputation

tionally the target attribute of the original data set. The choice of the repair algorithm is arbitrary, in principle any regression algorithm can be used here. In our experiments we used Random Forest [24], due to its superior accuracy, speed and robustness. Random Forest requires little to no tuning, which is very important when numerous models have to be developed without human assistance. Additionally, the Random Forest algorithm provides a heuristic for ranking attributes according to their importance. The IARI algorithm uses this heuristic for ordering the attributes.

It might seem counter-intuitive to include the target attribute in the set of predictors to impute an input attribute—it resembles a circular process. However, our goal is to repair a training set with help of any data we have. When the training set is fixed, a final model is trained and it can be applied to fresh data that were not used in the training process, so there is no circularity here. Moreover, results of our experiments demonstrate that including the target variable in the imputation process substantially increases the accuracy of the final model which is validated on data that were not used in the imputation process.

The IARI algorithm consists of two steps: initialization and main loop. During the initialization all attributes are split into two groups: those that contain no missing values (REPAIRED), and all others (TO_BE_REPAIRED). Assumed here is that the target attribute, $y$, contains no missing values so it falls into the REPAIRED group. Additionally, the set of attributes with missing values is ordered according to their importance. This is achieved in three steps. First, the training set is repaired with help of a simple imputation method which replaces missing values of continuous attributes by their mean values and missing values of discrete variables are replaced by their most frequent values. Second, a Random Forest model is built on the repaired training set to predict values of $y$. Finally, the model is applied to randomized out-of-bag samples to measure the importance of all attributes, as described in [18].

When the initialization step is finished, the algorithm enters the main loop which repairs attributes with missing values, one by one, in the order of their importance (from most to least important). To repair an attribute $x$, IARI creates a temporary training set which contains all attributes that are already repaired (including $y$) as predictors and $x$ as the target. All records where the value of $x$ is missing are
removed from this training set and, depending on the type of \( x \), a classification or regression variant of the Random Forest algorithm is used to model \( x \). Finally, the model is used to impute all missing values of \( x \) and \( x \) is moved from the TO_BE_REPAIRED to the REPAIRED set.

The pseudo-code of a generic version of the IARI algorithm is provided in Algorithm 3.2.

**Algorithm 3.2 Incremental Attribute Regression Imputation**

**Given:** A training set \( X \) with input attributes \( x_1, \ldots, x_n \), a target attribute \( y \), and a classification or regression algorithm \( ALG \)

**Initialization:**

\[
\begin{align*}
\text{for all attributes } x_i & \in X \text{ do} \\
N_{\text{missing}}[i] &= \text{Count}_\text{-missing}(x_i) \\
\text{Importance}[i] &= \text{ImportanceMeasure}(X, x_i, y)
\end{align*}
\]

\[
\text{REPAIRED} = y \cup \{\text{All attributes } x_i \text{ where } N_{\text{missing}}[i] = 0\}
\]

\[
\text{TO\_BE\_REPAIRED} = \{\text{All attributes } x_i \text{ where } N_{\text{missing}}[i] > 0\}
\]

**while** \( \text{TO\_BE\_REPAIRED} \neq \emptyset \) **do**

\[
\begin{align*}
\text{Repair\_Attribute} &= \text{SELECT}_X(TO\_BE\_REPAIRED, \text{Importance}) \\
\text{Repair\_Target} &= \text{Delete\_Missing\_Values}(\text{Repair\_Attribute}) \\
\text{Model} &= ALG\text{-train}(\text{REPAIRED}, \text{Repair\_Target})
\end{align*}
\]

\[
\text{for all records } A_j \in \text{Repair\_Attribute} \text{ do} \\
\quad \text{if } \text{is\_missing}(A_j) \text{ then} \\
\quad \quad A_j = ALG\text{-predict}(\text{REPAIRED}[j])
\]

\[
\text{end if}
\]

\[
\text{end for}
\]

\[
\text{REPAIRED} = \text{REPAIRED} \cup \text{Repair\_Attribute}
\]

\[
\text{TO\_BE\_REPAIRED} = \text{TO\_BE\_REPAIRED} \setminus \text{Repair\_Attribute}
\]

**end while**

**return** \( \text{REPAIRED} \)

### 3.3.1 Existing Imputation Algorithms

There are many ways of dealing with missing data when building a regression or classification model. Some of the most popular methods are:

**Complete Case Analysis (CCA):** This method simply ignores all records that
have missing values and selects only records with no missing values \cite{27,21}. When the percentage of complete records is relatively high and the data are missing at random or completely at random, this method does not affect model accuracy. However, if the amount of missing data is large, the prediction accuracy will be low (not enough complete cases), and when the data are missing not at random, then this method generates bias.

**Missing Indicator Variable (MIV):** This method uses a dummy variable as an indicator for missing values \cite{27}. For every variable that might be missing, a dummy variable is introduced, where the value of this dummy variable is 1 when the input variable is missing and 0 when the input variable is not missing. While this method is more efficient than the Complete Case Analysis, it can also create bias in the final model.

**Predictive Value Imputation (PVI):** PVI replaces missing values by some estimates of their values \cite{28}. In many cases the unconditional mean is used (the mean value of all non-missing values of the attribute) or a conditional mean (the mean of a specific group of records where the record with a missing value belongs to). The problem with this method is that the predictive values are always derived from the complete cases and that might introduce some bias. However, some additional mechanisms can be added to PVI which lower this bias. For example, PVI might use the conditional mean over the $k$-nearest neighbors of a record with a missing value, and then the bias can be limited by first imputing the data set with unconditional mean and then using the $k$-nearest neighbors on the completed data set to predict the values of the originally missing data. By counting the number of missing data in the neighbors, one can create a weighted average that incorporates the uncertainty of the measurements. There are several other methods to do single-value predictive imputation like *hot-deck imputation*, *cold-deck imputation* and *last observation carried forward*, where the data set is sorted on specific variables and when a missing value is encountered, the value is replaced by the value of its predecessor.

**Regression Imputation (RI):** Regression Imputation \cite{28} is a PVI variant where regression models (Support Vector Machines, Random Forests, etc.) are used to estimate the imputed value. One way is to build the models to estimate
3. MISSING VALUE ANALYSIS AND IMPUTATION

the missing values using the complete cases. However, it is usually better to also incorporate the non-complete cases by first imputing the missing values with a more simple imputation method (like the unconditional mean). In the first case (using only complete cases), there might be too few complete cases to generate good models, in the latter case there is a danger of bias by training the model with imputed (wrong) data.

**Multiple Imputation (MI):** This is a general imputation framework by Rubin et al. [20, 15, 22, 29]. The idea is to generate multiple versions of imputed (completed) data sets, which result in multiple models. Each model is then combined into a final predictor. The framework uses a single value imputation algorithm of choice and a random component that represents the uncertainty of the imputation. By creating multiple imputed data sets, the distribution of the imputed values will reflect the distribution of the already known values and therefore reduce bias. This method allows any non-deterministic imputation algorithm to be used. After imputing the data set several times, creating several copies, a model is being built for each complete data set. The results of each model are combined using Rubin’s Rules [20]. The combined result leads to less biased and more accurate predictions. One of the major advantages of MI is that it can be used with almost any imputation algorithm. Because of this, MI is not added in the comparison because each of the imputation algorithms can be wrapped with Multiple Imputation.

Most of the above methods can also be used for handling missing data at prediction time. The CCA method is here an obvious exception, but imputation or using dummy variables are valid ways to deal with missing values at prediction time. It should also be mentioned that in addition to the classical “off-line” scenario, where the training set is fixed and is not changing over time, some researchers were considering an “on-line” scenario, where the model is continuously updated while processing a stream of data [30].

In this chapter a novel strategy is proposed that uses regression models in an attribute wise algorithm to impute missing values in the training stage using the target attribute as one of the predictors. The proposed algorithm is compared
with commonly used imputation methods and an imputation method that also uses regression models: *Regression Imputation*.

### 3.3.2 Experimental Setup

To compare the existing algorithms with our approach we used five, very different, data sets from various Machine Learning Repositories: *Digits, Cover Type, House 16H, Page Blocks*, and *Concrete Compressive Strength*. For a complete overview of these data sets, see the public IARI repository, [31].

**Parameters**

In our experiments, a popular implementation of the Random Forest algorithm, that comes with the *Scikit-learn* Python package, [32], is used. The key learning parameter, the number of estimators, was set to 100, and the remaining parameters had default values.

For each data set several experiments are done with 75% of the attributes containing missing values and 25% of the attributes (randomly chosen) containing no missing values. The amount of missing values in the attributes with missing data, was set to 10, 20, 30, 40, 50, 60 percent and for each setup we run 20 experiments using different random seeds. In each experiment, the complete data set was split in a training (80%) and a test set (20%). The deletion of values, repairing the training set and final modeling was performed on the training set. The test set was used to estimate the accuracy of the final model. When removing values from the training set we used two strategies: “missing at random”, *MAR*, where values were removed uniformly at random, and “missing not at random”, *MNAR*, where only values bigger than the median value of the attribute, were removed uniformly at random.

**Performance Indicators**

Two aspects are measured for the quality of the imputation. First, the accuracy of the final model, that was trained on the repaired data set, is estimated with help
3. MISSING VALUE ANALYSIS AND IMPUTATION

of cross-validation. The accuracy was measured either by the ratio of correctly classified cases (in case of classification) or by the coefficient of determination, $R^2$, (in case of regression):

$$R^2 = 1 - \frac{\sum_i (p_i - y_i)^2}{\sum_i (y_i - \bar{y})^2}$$

where $y_i$ denotes the target value and $p_i$ the predicted value.

This score indicates how well the model fits the test data. The maximal value of $R^2$ is 1, meaning the perfect fit; values smaller than 1 reflect the error. Furthermore, the $R^2$ and accuracy scores of each data set are measured on final models that were developed with three algorithms: Random Forests, Support Vector Machines and Gradient Boosted Decision Trees. This is to demonstrate that the value of $R^2$ (or the accuracy score) depends on the regressor or classifier that is being used in the final modeling, and that it not always reflects the quality of the imputation itself.

Secondly, we measured the quality of the approximation of the imputed values. As all the imputed variables were numeric, we used the Root Mean Squared Error, $RMSE$, to measure the difference between the observed and imputed values:

$$RMSE = \sqrt{\frac{\sum (v_{observed} - v_{imputed})^2}{n}}$$

To make the comparison of results over various data sets meaningful, the attributes of all training sets are standardized by centering them around 0 and dividing by their standard deviations. As the last indicator of an algorithm’s performance, the execution time is measured. For bigger data sets the CPU time might be an issue to consider.

### 3.3.3 Results

For each data set, we performed 12 experiments: one for each of the percentage levels of missing values (from 10 to 60) combined with the type of missingness
3.3 Incremental Attribute Regression Imputation

(MAR or MNAR). Each experiment was repeated 20 times (with different random seeds) and the results were averaged. Additionally, for each reconstructed training set, we run three algorithms, Random Forests, Support Vector Machines and Gradient Boosted Decision Trees, to build the final models.

The results of our experiments, the accuracy of the final model ($R^2$ or the ratio of correctly classified cases) and the accuracy of imputation ($RMSE$), are presented in the following subsection. Each row contains averaged results of 20 runs of the same experiment with different random seeds. The amount of missing values and the type of missing values (MAR or MNAR) are shown as well. For the MNAR model we used the missing percentages 20%, 40%, and 60% as upper bounds for the percentage of missing values per attribute, but it was not always possible to delete that many values of the attribute due to the restriction of deleting only values bigger than the median. Let us note, that it may happen that the fraction of records with a value of an attribute bigger than its median might be arbitrarily small, e.g., when an attribute is almost constant. Moreover, in the results presented below, we show the average number of missing values taken over all attributes with missing values.

For the first three data sets (Cover Type, Digits and Houses 16H) we show the results from the Random Forest final model; for the remaining data sets and final model options please see Appendix [A].

Each table contains several columns. The first two columns contain information about the percentage of missing values and the type of missingness. The third column, Ref., contains the accuracy of the model trained on the original complete data set: either $R^2$ for regression problems or classification accuracy for classification problems. The following columns contain results of various imputation methods: Imputation by Mean, Imputation by Median, Imputation by Most Frequent, Predictive Value Imputation using 2-nearest neighbor over a data set imputed by the Mean, Regression Imputation using Random Forests and last but not least, our own algorithm: IARI. Entries in boldface are significantly better than all other entries with the same settings. The significance is tested using the $t$-test, with significance level $p = 0.05$. The absence of a bold entry in the row means that none of the results were significantly better than the others.
3. MISSING VALUE ANALYSIS AND IMPUTATION

Cover Type data set Results

In Table 3.2 and 3.3, the accuracy of the model (Accuracy Score) and the quality of imputation \((RMSE)\) are shown for the imputation algorithms on 40,000 instances of the Cover Type data set.

**Table 3.2**: Model Accuracy Score on the Cover Type data set with 40,000 instances using Random Forests

<table>
<thead>
<tr>
<th>Miss.%</th>
<th>Type</th>
<th>Ref.</th>
<th>Mean</th>
<th>Median</th>
<th>Freq.</th>
<th>PVI</th>
<th>NN</th>
<th>RI</th>
<th>IARI</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>MNAR</td>
<td>0.911</td>
<td>0.879</td>
<td>0.876</td>
<td>0.866</td>
<td>0.868</td>
<td>0.886</td>
<td>0.886</td>
<td>0.893</td>
</tr>
<tr>
<td>6</td>
<td>MNAR</td>
<td>0.911</td>
<td>0.864</td>
<td>0.860</td>
<td>0.868</td>
<td>0.868</td>
<td>0.886</td>
<td>0.886</td>
<td>0.881</td>
</tr>
<tr>
<td>10</td>
<td>MNAR</td>
<td>0.911</td>
<td>0.809</td>
<td>0.803</td>
<td>0.806</td>
<td>0.805</td>
<td>0.805</td>
<td>0.805</td>
<td>0.839</td>
</tr>
<tr>
<td>12</td>
<td>MNAR</td>
<td>0.911</td>
<td>0.678</td>
<td>0.656</td>
<td>0.657</td>
<td>0.663</td>
<td>0.663</td>
<td>0.663</td>
<td>0.693</td>
</tr>
<tr>
<td>10</td>
<td>MAR</td>
<td>0.911</td>
<td>0.893</td>
<td>0.899</td>
<td>0.900</td>
<td>0.900</td>
<td>0.900</td>
<td>0.900</td>
<td>0.907</td>
</tr>
<tr>
<td>20</td>
<td>MAR</td>
<td>0.911</td>
<td>0.874</td>
<td>0.887</td>
<td>0.886</td>
<td>0.883</td>
<td>0.883</td>
<td>0.883</td>
<td>0.899</td>
</tr>
<tr>
<td>40</td>
<td>MAR</td>
<td>0.911</td>
<td>0.834</td>
<td>0.859</td>
<td>0.858</td>
<td>0.845</td>
<td>0.845</td>
<td>0.845</td>
<td>0.878</td>
</tr>
<tr>
<td>60</td>
<td>MAR</td>
<td>0.911</td>
<td>0.776</td>
<td>0.824</td>
<td>0.822</td>
<td>0.787</td>
<td>0.799</td>
<td>0.799</td>
<td>0.847</td>
</tr>
</tbody>
</table>

**Table 3.3**: Imputation Quality (RMSE) of each Imputation Algorithm on the Cover Type data set with 40,000 instances

<table>
<thead>
<tr>
<th>Miss.%</th>
<th>Type</th>
<th>Mean</th>
<th>Median</th>
<th>Freq.</th>
<th>PVI</th>
<th>NN</th>
<th>RI</th>
<th>IARI</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>MNAR</td>
<td>0.755</td>
<td>0.762</td>
<td>0.774</td>
<td>0.755</td>
<td>0.745</td>
<td>0.721</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>MNAR</td>
<td>0.786</td>
<td>0.795</td>
<td>0.813</td>
<td>0.786</td>
<td>0.776</td>
<td>0.760</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>MNAR</td>
<td>0.848</td>
<td>0.852</td>
<td>0.867</td>
<td>0.847</td>
<td>0.838</td>
<td>0.791</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>MNAR</td>
<td>0.894</td>
<td>0.884</td>
<td>0.889</td>
<td>0.894</td>
<td>0.894</td>
<td>0.877</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>MAR</td>
<td>0.271</td>
<td>0.277</td>
<td>0.294</td>
<td>0.261</td>
<td>0.225</td>
<td>0.176</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>MAR</td>
<td>0.380</td>
<td>0.389</td>
<td>0.414</td>
<td>0.370</td>
<td>0.330</td>
<td>0.266</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>MAR</td>
<td>0.540</td>
<td>0.552</td>
<td>0.588</td>
<td>0.533</td>
<td>0.496</td>
<td>0.422</td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>MAR</td>
<td>0.661</td>
<td>0.676</td>
<td>0.718</td>
<td>0.658</td>
<td>0.630</td>
<td>0.564</td>
<td></td>
</tr>
</tbody>
</table>

**Table 3.4**: Execution time of Imputation Algorithms on the Cover Type data set with values 50% MAR in seconds

<table>
<thead>
<tr>
<th>Mean</th>
<th>Median</th>
<th>Freq.</th>
<th>PVI</th>
<th>NN</th>
<th>RI</th>
<th>IARI</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.03</td>
<td>0.11</td>
<td>0.48</td>
<td>61.47</td>
<td>381.75</td>
<td>119.12</td>
<td></td>
</tr>
</tbody>
</table>
3.3 Incremental Attribute Regression Imputation

From our test results we can observe that the maximum average number of MNAR values we can delete from each attribute is around the 12%. Which implies that approximately 88% of the data set is filled with values below or equal the median of each attribute (probably 0). In Table 3.4 the execution time for each algorithm is shown for the case of 50% values MAR, which is representative for all the tests on this data set. Our approach is not the fastest, Replace by Median, Replace by Mean and Replace by Most Frequent are almost instant while PVI, RI and IARI are more complex and take some time. The execution time is mostly dependent on the size of the data set and on the number of attributes, and not so much on the number of missing values.

Digits data set Results

In Table 3.5 the accuracy of the models created using the different imputed data sets as training set is shown.

<table>
<thead>
<tr>
<th>Miss.%</th>
<th>Type</th>
<th>Ref.</th>
<th>Mean</th>
<th>Median</th>
<th>Freq.</th>
<th>PVI</th>
<th>NN</th>
<th>RI</th>
<th>IARI</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>MNAR</td>
<td>0.972</td>
<td><strong>0.969</strong></td>
<td>0.967</td>
<td>0.953</td>
<td>0.967</td>
<td>0.967</td>
<td>0.968</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>MNAR</td>
<td>0.972</td>
<td><strong>0.966</strong></td>
<td>0.951</td>
<td>0.904</td>
<td>0.954</td>
<td>0.961</td>
<td>0.947</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>MNAR</td>
<td>0.972</td>
<td><strong>0.949</strong></td>
<td>0.923</td>
<td>0.815</td>
<td>0.934</td>
<td>0.932</td>
<td>0.944</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>MAR</td>
<td>0.972</td>
<td>0.964</td>
<td>0.963</td>
<td>0.961</td>
<td>0.967</td>
<td>0.968</td>
<td><strong>0.970</strong></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>MAR</td>
<td>0.972</td>
<td>0.954</td>
<td>0.957</td>
<td>0.952</td>
<td>0.959</td>
<td>0.960</td>
<td><strong>0.962</strong></td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>MAR</td>
<td>0.972</td>
<td>0.944</td>
<td>0.943</td>
<td>0.934</td>
<td>0.944</td>
<td>0.948</td>
<td><strong>0.953</strong></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Miss.%</th>
<th>Type</th>
<th>Mean</th>
<th>Median</th>
<th>Freq.</th>
<th>PVI</th>
<th>NN</th>
<th>RI</th>
<th>IARI</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>MNAR</td>
<td>0.608</td>
<td>0.649</td>
<td>0.752</td>
<td>0.566</td>
<td>0.565</td>
<td>0.479</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>MNAR</td>
<td>0.858</td>
<td>0.903</td>
<td>1.037</td>
<td>0.841</td>
<td>0.829</td>
<td>0.646</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>MNAR</td>
<td>0.974</td>
<td>0.994</td>
<td>1.103</td>
<td>0.960</td>
<td>0.963</td>
<td><strong>0.850</strong></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>MAR</td>
<td>0.380</td>
<td>0.399</td>
<td>0.511</td>
<td>0.299</td>
<td>0.316</td>
<td><strong>0.231</strong></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>MAR</td>
<td>0.537</td>
<td>0.564</td>
<td>0.721</td>
<td>0.470</td>
<td>0.472</td>
<td><strong>0.345</strong></td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>MAR</td>
<td>0.658</td>
<td>0.692</td>
<td>0.883</td>
<td>0.619</td>
<td>0.608</td>
<td><strong>0.451</strong></td>
<td></td>
</tr>
</tbody>
</table>
3. MISSING VALUE ANALYSIS AND IMPUTATION

In Table 3.6 the RMSE values for every imputation algorithm are presented for all the combinations of missing data percentage and missing data type. Our IARI approach outperforms the other imputation algorithms in most of the MAR cases with respect to the accuracy. In the MNAR cases our algorithm works well but imputation by Mean sometimes has a slightly better accuracy for the Random Forest model.

Houses 16H data set Results

In Table 3.7 the $R^2$ scores are shown and in Table 3.8 the RMSE results are shown for the imputation algorithms on the Houses 16H data set.

**Table 3.7:** Model Accuracy Score ($R^2$) on the Houses data set using Random Forests

<table>
<thead>
<tr>
<th>Miss.%</th>
<th>Type</th>
<th>Ref.</th>
<th>Mean</th>
<th>Median</th>
<th>Freq.</th>
<th>PVI</th>
<th>NN</th>
<th>RI</th>
<th>IARI</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>MNAR</td>
<td>0.636</td>
<td>0.604</td>
<td>0.598</td>
<td>0.580</td>
<td>0.606</td>
<td>0.603</td>
<td>0.617</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>MNAR</td>
<td>0.636</td>
<td>0.534</td>
<td>0.491</td>
<td>0.485</td>
<td>0.511</td>
<td>0.520</td>
<td>0.531</td>
<td></td>
</tr>
<tr>
<td>49</td>
<td>MNAR</td>
<td>0.636</td>
<td>−0.277</td>
<td>−0.287</td>
<td>−0.545</td>
<td>−0.405</td>
<td>−0.171</td>
<td>−0.450</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>MAR</td>
<td>0.636</td>
<td>0.604</td>
<td>0.599</td>
<td>0.586</td>
<td>0.610</td>
<td>0.598</td>
<td>0.620</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>MAR</td>
<td>0.636</td>
<td>0.544</td>
<td>0.533</td>
<td>0.511</td>
<td>0.552</td>
<td>0.521</td>
<td>0.590</td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>MAR</td>
<td>0.636</td>
<td>0.423</td>
<td>0.402</td>
<td>0.375</td>
<td>0.458</td>
<td>0.414</td>
<td>0.536</td>
<td></td>
</tr>
</tbody>
</table>

**Table 3.8:** Imputation Quality (RMSE) of each Imputation Algorithm on the Houses data set

<table>
<thead>
<tr>
<th>Miss.%</th>
<th>Type</th>
<th>Mean</th>
<th>Median</th>
<th>Freq.</th>
<th>PVI</th>
<th>NN</th>
<th>RI</th>
<th>IARI</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>MNAR</td>
<td>0.486</td>
<td>0.517</td>
<td>0.709</td>
<td>0.485</td>
<td>0.428</td>
<td>0.342</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>MNAR</td>
<td>0.785</td>
<td>0.801</td>
<td>1.007</td>
<td>0.787</td>
<td>0.753</td>
<td>0.587</td>
<td></td>
</tr>
<tr>
<td>49</td>
<td>MNAR</td>
<td>0.956</td>
<td>0.925</td>
<td>1.134</td>
<td>0.955</td>
<td>0.954</td>
<td>0.927</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>MAR</td>
<td>0.386</td>
<td>0.395</td>
<td>0.542</td>
<td>0.370</td>
<td>0.328</td>
<td>0.280</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>MAR</td>
<td>0.545</td>
<td>0.558</td>
<td>0.764</td>
<td>0.535</td>
<td>0.487</td>
<td>0.412</td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>MAR</td>
<td>0.609</td>
<td>0.685</td>
<td>0.925</td>
<td>0.665</td>
<td>0.625</td>
<td>0.531</td>
<td></td>
</tr>
</tbody>
</table>
3.4 Attribute Selection and Sorting Methods

For the IARI algorithm the most important attributes are selected to be imputed first by using the out-of-bag samples provided by the random forest algorithm. However, the question is if this is the best possible order of imputation. Comparing the scores of IARI using this attribute ordering with the scores of IARI using the exact opposite attribute ordering (least important attribute first) shows that the results are only slightly worse for the latter case. This implies that the order of attribute repair does matter, but is not a key factor in the algorithm.

3.4.1 Greedy Model Accuracy Selection

Another sorting or attribute selection method would be to repair each attribute first, using the already repaired attributes, and select the attribute that improves the accuracy of the model the most. This selected attribute is then added to the training set and the procedure is repeated till all remaining attributes do not improve the accuracy any further. This greedy approach leads to an algorithm where one can decide to stop imputing the attributes once the model does not improve any longer. Effectively, giving an algorithm that not only repairs the attributes of the data set but also selecting which attributes are useful to impute and which attributes are not improving our model and so might be better left out.

Running this greedy version of our algorithm on the Digits data set we get the $R^2$ results of Table 3.9.

The greedy version of IARI uses on average only 22 of the 64 attributes that are available. It is interesting to see that the results of our greedy approach are actually worse than the results of our non-greedy approach for the Digits data set. The reason, is most likely that too many attributes are ignored (discarded) by the greedy approach. It can be the case that adding an attribute to our model might not improve the model’s accuracy immediately, but adding this attribute in combination with another attribute might still improve the model. This can clearly be seen in the following example where a model is trained on the function
3. MISSING VALUE ANALYSIS AND IMPUTATION

Table 3.9: Model Accuracy Score ($R^2$) for the greedy IARI approach on the Digits data set

<table>
<thead>
<tr>
<th>Miss.%</th>
<th>Type</th>
<th>Ref.</th>
<th>IARI</th>
<th>GREEDY</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>MNAR</td>
<td>0.964</td>
<td><strong>0.961</strong></td>
<td>0.954</td>
</tr>
<tr>
<td>16</td>
<td>MNAR</td>
<td>0.964</td>
<td><strong>0.953</strong></td>
<td>0.949</td>
</tr>
<tr>
<td>23</td>
<td>MNAR</td>
<td>0.964</td>
<td><strong>0.950</strong></td>
<td>0.912</td>
</tr>
<tr>
<td>25</td>
<td>MNAR</td>
<td>0.964</td>
<td><strong>0.926</strong></td>
<td>0.907</td>
</tr>
<tr>
<td>27</td>
<td>MNAR</td>
<td>0.964</td>
<td><strong>0.915</strong></td>
<td>0.908</td>
</tr>
<tr>
<td>10</td>
<td>MAR</td>
<td>0.964</td>
<td>0.956</td>
<td>0.952</td>
</tr>
<tr>
<td>20</td>
<td>MAR</td>
<td>0.964</td>
<td>0.951</td>
<td>0.951</td>
</tr>
<tr>
<td>30</td>
<td>MAR</td>
<td>0.964</td>
<td><strong>0.955</strong></td>
<td>0.943</td>
</tr>
<tr>
<td>40</td>
<td>MAR</td>
<td>0.964</td>
<td><strong>0.952</strong></td>
<td>0.941</td>
</tr>
<tr>
<td>50</td>
<td>MAR</td>
<td>0.964</td>
<td><strong>0.948</strong></td>
<td>0.935</td>
</tr>
<tr>
<td>60</td>
<td>MAR</td>
<td>0.964</td>
<td><strong>0.934</strong></td>
<td>0.921</td>
</tr>
</tbody>
</table>

Table 3.10: Model Accuracy Score ($R^2$) for the greedy IARI approach on the Houses data set

<table>
<thead>
<tr>
<th>Miss.%</th>
<th>Type</th>
<th>Ref.</th>
<th>IARI</th>
<th>GREEDY</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>MNAR</td>
<td>0.636</td>
<td>0.630</td>
<td><strong>0.632</strong></td>
</tr>
<tr>
<td>8</td>
<td>MNAR</td>
<td>0.636</td>
<td>0.617</td>
<td><strong>0.624</strong></td>
</tr>
<tr>
<td>8</td>
<td>MNAR</td>
<td>0.636</td>
<td>0.585</td>
<td><strong>0.605</strong></td>
</tr>
<tr>
<td>8</td>
<td>MNAR</td>
<td>0.636</td>
<td>0.531</td>
<td><strong>0.574</strong></td>
</tr>
<tr>
<td>8</td>
<td>MNAR</td>
<td>0.636</td>
<td>−0.450</td>
<td><strong>0.016</strong></td>
</tr>
<tr>
<td>10</td>
<td>MAR</td>
<td>0.636</td>
<td>0.627</td>
<td><strong>0.635</strong></td>
</tr>
<tr>
<td>20</td>
<td>MAR</td>
<td>0.636</td>
<td>0.620</td>
<td><strong>0.631</strong></td>
</tr>
<tr>
<td>30</td>
<td>MAR</td>
<td>0.636</td>
<td>0.608</td>
<td><strong>0.624</strong></td>
</tr>
<tr>
<td>40</td>
<td>MAR</td>
<td>0.636</td>
<td>0.590</td>
<td><strong>0.614</strong></td>
</tr>
<tr>
<td>50</td>
<td>MAR</td>
<td>0.636</td>
<td>0.567</td>
<td><strong>0.599</strong></td>
</tr>
<tr>
<td>60</td>
<td>MAR</td>
<td>0.636</td>
<td>0.536</td>
<td><strong>0.580</strong></td>
</tr>
</tbody>
</table>

$y = OR(x_1, XOR(x_2, x_3))$. If the model is trained using only $x_1$, the final accuracy will be 75%. When adding $x_2$ to the training set (after fixing possible missing values) the accuracy of the model will not improve. However, by also adding $x_3$ to the training set the model’s accuracy will increase to 100% (under the assumption that there are sufficient samples to learn from).

This simple example tells us that combinations of attributes might be valuable
even if all the single attributes are not giving any improvement on the final model. In the case of the Digits data set, this might be the case since each attribute only stands for a single pixel, and combinations of pixels create the information needed to see what digit the image represents, while single pixels might not give any information at all.

The greedy algorithm is also evaluated on the Houses 16H data set, which seems a more suitable data set for such an approach. The data set has 16 attributes and the greedy algorithm uses on average 13 of them. The $R^2$ scores of the original IARI and the greedy modification can be seen in Table 3.10. In this case the greedy algorithm performs slightly better than the original IARI algorithm. It clearly depends on the data set and the dependencies between the attributes if the greedy attribute selection works or not.

### 3.4.2 Greedy Imputation Quality Selection

Another possibility to optimize the IARI algorithm is to select the features to be repaired on the repairability of each attribute. With repairability we mean how well a specific attribute can be repaired. This can be measured by the out-of-bag error provided by the Random Forest models used to repair each attribute with. The main idea behind this approach is that if the attributes are repaired in an order of attributes that can be repaired as good as possible first, it might be possible to repair the remaining attributes even more accurately and as such, the final model will have a lower $RMSE$. In Figure 3.4 the $RMSE$ of the IARI algorithm and the repairability selection variant of IARI (named REPAIR) are presented. The REPAIR algorithm has a slight improvement in the RMSE score, but the modified algorithm also takes more time due to the $n \times (n - 1)$ models it trains.

### 3.5 Conclusions

A new concept to analyze and visualize missing value patterns in data sets is proposed and it is shown that with a greedy method called MMP-finder, a data
3. MISSING VALUE ANALYSIS AND IMPUTATION

Figure 3.4: Imputation Quality (RMSE) on the Concrete data set

Figure 3.5: Imputation Quality (RMSE) on the Page data set
set with missing values can be analyzed. Using the concept of *k-Monotone Mixture Patterns*, a better in-depth understanding of the underlying patterns of missing values or unique values can be obtained.

Furthermore, a missing value generator is developed and made publicly available to make it possible to test and compare different algorithms on a wide set of both generated and natural data sets. The proposed missing value generator can be used as a tool to generate benchmark data sets for algorithms that handle certain kinds of missing values, giving insight in what kind of situations an algorithm might work well and in what situations certain algorithms would not perform well. For future work, it would be interesting to incorporate various mechanisms of missing values into the generator to also cover MNAR situations. Another interesting extension to the proposed analysis algorithm would be to include expert knowledge and assumptions to come up with the most likely partition of monotone patterns.

In addition, a novel algorithm, IARI, is proposed for imputing missing values into training sets. IARI can handle both regression and classification problems. The key advantage of IARI over other imputation methods is the superior accuracy of the final models which are trained on the repaired training sets, and more accurate reconstruction of missing values. On the other hand, IARI requires much more computing resources than its alternatives: 2-3 orders of magnitude. Fortunately, the main algorithm behind IARI, Random Forest, can be efficiently distributed along multiple nodes, significantly reducing the real (wall clock) computation time.

In principle, IARI is a generic algorithm which can be configured in various ways by changing the measure of importance of attributes, ordering of attributes, and the base algorithm used for imputation. Also the initialization step, where only attributes with no missing values are used as a starting set of predictors, can be modified: sometimes adding to this set several attributes with just a few missing values and removing incomplete records from it, lead to better results.

During our experiments with IARI, it was noticed that sometimes a simple imputation method may lead to better results than those obtained by IARI. This
happens in case of the *Digits* data set, where values were removed “not at random”, see the *IARI repository* \[31\] for additional details. As expected, the quality of IARI approximations of missing values was always significantly better than those obtained by imputing means, but surprisingly, the opposite holds for the quality of the corresponding final models. This could be caused by the nature of the classification problem and the fact that Random Forest is not suitable for image classification. Almost in all other cases, the IARI algorithm outperforms other imputation methods: both in terms of the accuracy of imputation and the accuracy of the final model. In most real world cases it is difficult to determine how well a certain imputation algorithm will work. The quality of imputation depends a lot on the data set and the reason of why values are missing. However, when little is known about a data set, the IARI algorithm is a good choice to start with.

Both the analysis and visualization methods and the IARI algorithm are publicly available on Github in the *IARI repository* \[31\] and *MisVis repository* \[33\].