INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

University of Alberta

Novel Framework for Imputation of Missing Values in Databases

by

Alireza Farhangfar 

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the

Master of Science

Department of Electrical and Computer Engineering

Edmonton, Alberta

Spring 2005

University of Alberta

Library Release Form

Name of Author: *Alireza Farhangfar*

Title of Thesis: *Novel Framework for Imputation of Missing Values in Databases*

Degree: *Master of Science*

Year this Degree Granted: *2005*

Signature

Dedication

To my family, especially my parents, for their continuous support. They have put incredible amount of effort to help me with my education, and with finding the proper way of living and studying. I especially thank them for continuing words of encouragement.

Abstract

Many of the industrial and research databases are plagued by missing values problem. One of common ways to cope with this problem is to perform imputation (filling in) of the missing values through variety of statistical and machine learning (ML) procedures. This study concentrates on performing experimental comparison between several algorithms for imputation of missing values, which range from simple statistical algorithms such as mean and hot deck imputation to imputation algorithms that work based on application of inductive ML algorithms. The thesis also proposes a new framework that can be used to improve the accuracy of existing imputation method while maintaining the same asymptotic computational complexity. Extensive experimental test were performed and the results show that a significant improvement of imputation accuracy can be achieved by applying the proposed framework, and that the accuracy of the framework based methods is, on average, the highest among the considered methods.

Acknowledgements

I would like to thank all the people who helped me to finish this thesis and to continue my studies. Especially I appreciate my supervisors Dr. Lukasz Kurgan and Dr. Witold Pedrycz whose experience and continuous support helped me, over the last few years, to become a successful researcher. Their understanding and encouragement for my research directions, which resulted in establishing a friendly and strong advisor-student relationship, motivated me to continue my research and target ambitious goals.

I would like to thank the members of my thesis committee, Dr. Marek Reformat, and Dr. Jozef Szymanski. I also want to thank Dr. Horacio Marquez, the chair of the Electrical and Computer Engineering department for his numerous advises and help during my M.Sc. program.

Table of Contents

1. Introduction 1
   1.1 Goals of the Research 4
2. Background and Definitions 7
   2.1 Types of Data 7
   2.2 Background 8
   2.3 Definitions 9
      2.3.1 Training Set 9
      2.3.2 Class 10
      2.3.3 Learning the classification rules 10
   2.4 Inductive learning 11
      2.4.1 Machine learning 12
      2.4.2 Accuracy Test 13
      2.4.3 Verification Test 13
      2.4.4 Classification 15
3. Methods for Handling Missing Data 17
   3.1 Single and Multiple Imputation Methods 17
   3.2 Data Driven, Model Based, and ML Based Imputation Methods 19
      3.2.1 Data Driven Imputation Methods 19
      3.2.2 Model Based Imputation Methods 20
      3.2.3 Machine Learning Based Methods 20
   3.3 History and Motivation 23
4. Proposed Framework 25
   4.1 Introduction 25
   4.2 Relevant Imputation Methods 26
      4.2.1 Single Imputation Methods 27
      4.2.2 Multiple Imputation methods 30
   4.3 Detailed Description of the Proposed Framework 32
      4.3.1 Mean Pre-Imputation Module 32
      4.3.2 Confidence Intervals Module 36
      4.3.3 Boosting 41
5. Experiments and Results 46
   5.1 Experimental Analysis of Imputation Methods 46
      5.1.1 Experimental Setup 47
      5.1.2 Relevant Database Characteristics 48
      5.1.3 Comparison of the data imputation methods 49
      5.1.4 Discussion 56
   5.2 Missing Data Imputation Using the Proposed Framework 57
      5.2.1 Experimental Setup 59
      5.2.2 Framework Modules Evaluation 62
      5.2.3 Experimental Comparison with Other Imputation Methods 68
      5.2.4 Analysis of Experimental Complexity 71
6. Summary and Conclusions 78
References 80

List of Tables

Table 1.Summary of the imputation methods used in this study 31
Table 2.Description of the databases used in the experiments 60
Table 3. Running time of the eight imputation methods, for the fourteen databases and the six levels of missing values 75

List of Figures

Figure 1. A database containing missing values 8
Figure 2. The grades of the students 8
Figure 3.Training set related to student's grade dataset 10
Figure 4. Learning classification rules from a database 11
Figure 5. Classification task 15
Figure 6. Example of a decision tree 16
Figure 7. Flow of operations in multiple imputation 19
Figure 8. ML imputation process 23
Figure 9.Structure of the proposed framework 25
Figure 10. Pseudo code of the mean pre-imputation procedure 34
Figure 11. Pseudo code for the procedure of computing the confidence intervals 38
Figure 12. Pseudo code of the boosting procedure for the Naïve Bayes imputation method as the base method 44
Figure 13. Accuracy against amount of missing values 51
Figure 14. Slope of the accuracy trend for different amounts of missing values 51
Figure 15. Normalized rank of the average imputation accuracy versus the number of examples 53
Figure 16. Normalized rank of the average imputation accuracy versus the Boolean characteristic 53
Figure 17. Normalized rank of the average imputation accuracy vs. the average number of examples / class 54
Figure 18. Normalized rank of the average imputation accuracy versus the number of attributes 54
Figure 19. Average sensitivity versus the amount of missing values 55
Figure 20. Average specificity versus the amount of missing values 55
Figure 21. Imputation procedure including introducing missing values and imputing the missing values 59
Figure 22. Experimental evaluation of the proposed framework and its components 61
Figure 23. Improvement in imputation accuracy for hot deck base imputation method . 63
Figure 24. Difference in imputation accuracy between the hot deck with and without the framework (FHD - HD) for six levels of missing values and for fifteen databases . 64
Figure 25. Improvement in imputation accuracy for Naïve Bayes ML base imputation method 65
Figure 26. Difference in the accuracy of imputation between the Naïve-Bayes ML imputation method with and without the framework (FNB - NB) for six levels of missing values and for fifteen databases 66
Figure 27. Accuracy of imputation using framework with hot deck and standalone hot deck 67
Figure 28. Accuracy of imputation using framework with Naive-Bayes ML and standalone Naive-Bayes ML method 67
Figure 29. Summary of imputation accuracy results for the considered eight imputation methods 69
Figure 30. Accuracy of imputation using the framework with Naive-Bayes against other imputation methods 71

Figure 31. Accuracy of imputation using the framework with Hot deck against other imputation methods 71
Figure 32. Runtime against the size of the database for the FNB and NB imputation methods and for 10% and 60% of missing values 72
Figure 33. Run time against the size of the database for Hot deck imputation with and without the framework, for 10% and 60% of missing values 74

1. Introduction

Many of industrial and research databases are characterized by an unavoidable problem of incompleteness in terms of missing values. A variety of reasons, such as manual data entry procedures, incorrect measurements and equipment errors result in this serious deficiency. In many domains it is not uncommon to encounter databases that have up to or even above 50% of their entries missing. For example, an industrial instrumentation maintenance and test database maintained by Honeywell which has over 50% of missing information, despite regulatory requirements for data collection, is described in [44]. Another domain plagued by missing values problem is medicine, where it is not uncommon that almost every patient-record lacks some values, and almost every attribute used to describe patient's records is lacking values for some patient-record [18]. For example, a medical database describing patients with cystic fibrosis that has over 60% of its entries missing was described in [43]. One of the reasons why medical databases are so heavily exposed is that most medical data are collected as a byproduct of patient care activities, rather than for an organized research protocol [18].

Missing values usually make it difficult for analysts to perform data analysis tasks. Three types of problems are associated with missing values: loss of efficiency, complication in handling and analyzing the data, and bias resulting from differences between missing and complete data [3]. We note that although some of the data analysis methods can cope with databases that have missing values on their own, many others require the analyst to provide complete database to perform analysis. Standard statistical software works only with complete data, or uses very basic imputation methods [44]. Other data processing

packages, such as visualization and modeling software, often use and display only the complete records, or map missing values to an arbitrary fixed value, e.g. -1 or 999999, leading to distortion of the presented results. Imputation is also invaluable in cases when the data needs to be shared, and the individual users may not have resources to deal with the incompleteness [47] [60].

There are two general approaches to deal with the problem of missing values: they can be ignored (removed), or imputed (filled in) with new values. The first solution is applicable only in case where only a small amount of data is missing. Since in many cases databases contain substantial amount of missing data, the focus of this study is on the missing data imputation methods. A number of different imputation methods are reported in the literature. Traditional methods use statistical approaches, and include simple algorithms, such as mean and hot deck imputation, and complex methods, such as regression based imputation and Expectation-Maximization (EM) algorithm based imputation. In recent years a new family of imputation methods, which uses Machine Learning (ML) algorithms, was proposed. In general, development of new methods was driven by a need to improve accuracy of the imputation. Early methods were very simple and computationally inexpensive, while newer methods use more complex procedures, which improve the quality of imputation, but also require very expensive computations. At the same time, a rapid and increasing growth trend in the database size is documented. Recently published results of a 2003 survey on the largest and most heavily used commercial databases show that the average size of Unix databases experienced a 6-fold,

and Windows databases a 14-fold, increase compared to year 2001; large commercial databases now average 10 billion data points [75].

This study first performs experimental comparison of several algorithms for imputation of missing values, which range from simple statistical algorithms like mean and hot deck imputation to imputation algorithms that work based on application of inductive ML algorithms. Three major families of ML algorithms[19], such as probabilistic algorithms (e.g. Naïve Bayes), decision tree algorithms (e.g. C4.5), and decision rule algorithms (e.g. CLIP4), are used to implement the ML based imputation algorithms. The study also proposes a new framework that can be used to improve the quality of existing imputation methods, called *base* imputation methods. The base imputation methods are selected based on our comparative study. The developed framework is characterized by the following advantages:

- It **improves the accuracy of imputation** when compared to accuracy of a base imputation method.
- Application of the framework to a base imputation method **does not worsen the overall asymptotic computational complexity**, when compared to complexity of the base imputation method. We note that for some base methods, it can even result in improving their running time.
- **Many of imputation methods can be used as the base imputation methods**. The proposed framework can be applied to both statistical and ML based imputation methods.

Extensive experimental results presented in this thesis show that application of the proposed framework to the base imputation methods gives substantial improvement in accuracy of imputation. The experimental results show that using the proposed framework with very simple imputation methods, such as hot deck, gives imputation accuracies that surpass the quality of results generated by advanced statistical imputation methods, while having the same computational cost. As an example, the proposed framework was applied to a linear complexity imputation method, i.e. a ML based imputation that uses Naïve Bayes algorithm. The resulting imputation method was also linear, and the imputed missing values had an accuracy higher than that of any other considered imputation method, including complex statistical methods.

Following, the second chapter discusses the background and definitions related to missing data imputations. Existing imputation methods are reviewed in chapter 3. Chapter 4 describes and discusses structure of the proposed framework. Chapter 5 experimentally compares performance of several imputation methods and the results of this comparison are used to choose the base imputation methods for the proposed framework. The second part of chapter 5 reports experimental results performed with the proposed framework, and provides extensive comparison with existing imputation methods. The study ends with summary and conclusions.

1.1 Goals of the Research

It is not uncommon to encounter databases that have close to a half of information missing, making them almost unusable to perform data analysis tasks. One of the common ways to cope with this problem is to perform imputation (filling in) of the

missing values. Main aim of the existing and newly developed imputation methods is to improve accuracy. Early methods were computationally inexpensive and characterized by relatively poor performance, while newer methods improve the quality of imputation, but with the cost of higher computational complexity. The first part of the study concentrates on performing experimental comparison of several algorithms for imputation of missing values, which range from simple statistical algorithms like mean and hot deck imputation to imputation algorithms that work based on application of inductive ML algorithms. Three major families of ML algorithms, i.e. probabilistic, decision trees, and decision rule algorithms are used to implement the ML based imputation algorithms. Due to the recent rapid increasing growth trend in the database size, a new methodology, which is both efficient and accurate, is necessary. To this end, this study proposes a new framework that can be used to improve accuracy of existing imputation method. The framework's application to imputation methods should on average result in significant improvement of imputation accuracy, while maintaining the same asymptotic computational complexity. The study also presents a comprehensive review of relevant missing data imputation methods.

Extensive experimental tests were performed to compare accuracy, running time, and asymptotic complexity of application of the proposed framework to two imputation methods with six other state of the art imputation methods. The results show that a significant improvement of imputation accuracy can be achieved by applying the proposed framework, and that the accuracy of the framework based methods is, on average, the highest among the considered methods. We stress that application of the framework to a low quality single imputation method results in a method that has

imputation accuracy comparable to accuracy of advanced multiple imputation methods, while application to a quality single imputation method results in imputation accuracy that is superior to other imputation methods. We also show, both theoretically and experimentally, that application of the proposed framework has linear complexity, and therefore does not change asymptotic complexity of the associated base imputation method.

The contents of this thesis covers the materials which are published in [28] and [27]. The first part of studies, which is experimental comparison between several algorithms to impute missing values is mentioned in [28] and is the topic of the chapters 3 and 5.1. Also the idea of applying the framework to improve the accuracy of imputation is discussed in [27] which is the topic of chapters 4 and 5.2.

2. Background and Definitions

This chapter provides an overview of the definitions for the concepts used in this study including training set, class, and learning the classification rules. It also introduces different types of data and defines the inductive learning concept, machine learning algorithms, and classifications. The verification test and accuracy test, which are used in evaluating the performance of the imputation methods, are described.

2.1 Types of Data

There are four types of data that appear in databases: nominal, ordinal, interval, and ratio scales. They are categorized into two groups: categorical and continuous data; nominal and ordinal scales are categorical data.

Categorical data with unordered scales are called nominal scales. In grades dataset, see Figure 2, Name is an example of a nominal scale. Categorical data with ordered scales are called ordinal scale. Rank is an example of an ordinal scale. Continuous data are a type of raster data that are quantitative (measuring a characteristic) and have related continuous values, such as the assignments grades in Figure 2. The reason for us to distinguish between different types of data is that some data analysis methods can only deal with specific type of data; for instance some methods may not be applicable to the databases that contain both categorical and continuous data. Some ML algorithms can only deal with the categorical data, and therefore continuous data should be discretized prior to applying them.

2.2 Background

Databases consist of one or multiple tables, where columns describe attributes (features), and rows describe records (examples or data points). Figure 1 shows a typical database, which consists of five attributes, and where some of them contain missing values denoted by "?".This study concentrates on imputation procedures for categorical attributes. We note that the two main application areas of missing data imputation procedures are concerned with equipment maintenance databases [44], and survey data [29][59][61], both of which use discrete data. Some of the missing data imputation algorithms are supervised, i.e. they require so called class attribute. They impute missing values one attribute at the time by setting it to be the class attribute, and use data from the remaining attributes to generate a classification model, which is used to perform imputation.

Attribute 1	Attribute 2	Attribute 3	Attribute 4	Attribute 5
data	?	data	data	data
?	data	data	?	data
data	data	data	?	data
data	data	data	data	data
...	...	...	...	...

Figure 1. A database containing missing values

For example Figure 2 shows grades of the students for a certain course. In this table the first attribute is the name of the students, the second attribute is the final mark of the students, the third attribute is the assignment marks and the forth attribute is class attribute, which classifies the students into two groups, pass and fail.

Name	Final mark	Assignments	Status
Kate	A	95%	Pass
Sara	C	30%	Fail
Ali	?	60%	Pass
George	?	50%	Pass
Paul	?	35%	Fail
Emily	?	70%	Pass
John	B	75%	Pass
Jim	A	90%	Pass

Figure 2. The grades of the students

Three different modes that lead to introduction of missing values can be distinguished: missing completely at random (MCAR), missing at random (MAR), and not missing at random (NMAR) [44] [47].

- The *MCAR* mode applies when the distribution of a record having a missing value for an attribute does not depend on either the observed data or the missing data. This mode usually does not hold for non-artificial databases. For example, a student's final grade is missing, and this does not depend on his or her status or final grade.
- The *MAR* mode, where the distribution depends on the data, but does not depend on the missing data itself, is assumed by most of the existing methods for missing data imputation [68], and therefore it is also assumed in this study. For example, student's final mark is missing, and this does depend on his status, but not on the final grade. In case of the MCAR mode the assumption is that the distribution of missing and complete data are the same, while for MAR mode they are different, and the missing data can be predicted by using the complete data [47].
- The third, *NMAR*, mode, where the distribution depends on the missing values, is rarely used in practice. For example, student's final grade is missing, and this does depend on the final grade in terms that grades in a special range are all missing.

Below we define the terminology and concepts of ML, since they are used to develop the imputation procedures presented in this work.

2.3 Definitions

2.3.1 Training Set

Let $A = \{A_1,...,A_n\}$ be a set of attributes with domains $Dom_1,...,Dom_n$. A training set is a table over A. As an example Figure 3 shows a training set related to the grades of the students.

Name	Final mark	Assignments	Status
Kate	A	95%	Pass
Sara	C	30%	Fail
John	B	75%	Pass
Jim	A	90%	Pass

Figure 3.Training set related to student's grade dataset

In this table the first attribute is the name of the student, the second is the final mark, the third is the assignment mark and the forth is the status of the student. The domain Dom_1 is the set {Kate, Sara, John, Jim}, Dom_2={A, B, C}, Dom_3={95%, 30%, 75%, 90%} and Dom_4={Pass, Fail}.

2.3.2 Class

A class C_i is a subset of the training set S, consisting of all objects that satisfy the class condition $cond_i$:

$$C_i = \{o \in S \mid cond_i(o)\}$$

Objects that satisfy the condition $cond_i$ are positive records or instances of class C_i. The records outside of this subset of the training set are negative records.

2.3.3 Learning the classification rules

Learning classification rules means that the system has to find the rules that predict the class from the other attributes. Hence, first the user has to define the conditions for each class, and thus partition the S into subsets $C_1,...,Cn$ as illustrated in Figure 4. Then the

data mine system has to construct descriptions $D_1, \ldots, D_n$ for these classes, which results in determining the classes for the records.

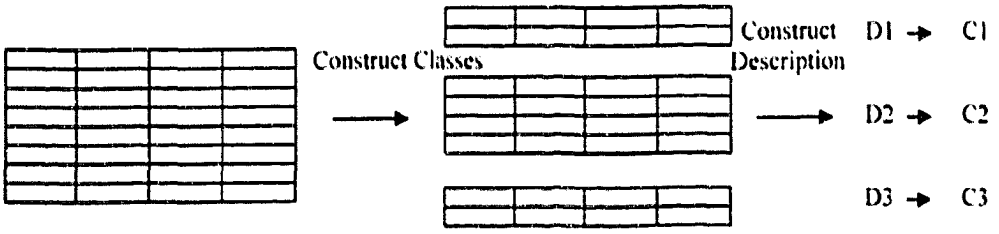


Figure 4. Learning classification rules from a database

2.4 Inductive learning

Human and other intelligent entities (cognitive systems) attempt to understand their environment by using a simplification of this environment called a model. The creation of such a model is called *inductive learning*. During the learning phase, the cognitive system observes its environment and recognizes similarities among objects and events in this environment. It groups similar objects in classes and constructs rules that predict the behavior of the inhabitants of such a class.

Two learning techniques are of special interests, supervised and unsupervised learning. In supervised learning, an external teacher defines classes and provides the cognitive system with examples of each class. The system has to discover common properties in the examples for each class, which are called class description. This technique is also known as learning from examples. A class, together with its description forms a classification rule, such as "if <description> then <class>" that can be used to predict the class of previously unseen objects.

In unsupervised learning the system has to discover the classes by itself, based on common properties of objects. Hence, this technique is also known as learning from observation.

2.4.1 Machine learning

The automation of inductive learning processes has been extensively researched in ML [42][45][51].

A ML process consists of two phases. The learning phase, in which a ML system analyzes the data and generates rules by finding similarities among the data, and the validation phase, in which the generated rules are verified by computing a performance evaluation function on new set of data. The training data set consists of M training data pairs (records):

$$S = \{(x_i, c_j) \mid i = 1,....,M; j = 1,....,C\}$$

where: x_i is n-dimensional pattern vector, whose components are called attributes, and c_j is a known class.

The ML algorithm's role is to search the space of possible hypotheses to discover the best estimate of the mapping function f, such that $c_j = f(x), j = 1,....,C.$. For the search to be successful the assumption has to be made that the attributes represent only properties of the records but not the relationships between the records. A ML algorithm generates hypotheses by finding common attributes and their values for records representing each class. Then, the generated hypotheses are applied to new records to predict their class membership.

One of the issues in ML is presence of noise in the data. The noise can be present in the attributes and/or in the class descriptions (false records). Only some of ML algorithms are noise-tolerant, which means that they can generate rules that do not cover noisy records. Another issue is generalization and specialization of the generated rules. A general rule covers more records, and thus might perform better on unseen data than a specific rule. A record is covered by a rule when it satisfies all conditions of the *if* part of the rule. In cases where the number of the generated hypotheses is excessively large an algorithm has to choose a subset of them [19] by means of:

- Heuristics
- Minimum description length principle
- Background knowledge about the domain
- Reasoning from first principles (like laws of physics, mathematical)
- Decisions made by the user, based on his/her knowledge of the problem.

2.4.2 Accuracy Test

The accuracy test quantifies quality of a learning method such as missing data imputation procedure. An accuracy test is defined as:

$$Accuracy = \frac{Correct\_imputation}{Total}$$

Where *Correct_imputation* indicates the number of imputed values, which are the same as the original values, and the *Total* is the total number for missing values.

2.4.3 Verification Test

When two outcomes (positive and negative) of the imputation are possible the following three evaluation criteria, collectively called as a verification test, can be used for measuring the effectiveness of the imputation procedure. In this case we have four possibilities, as shown below

	Imputation positive	Imputation negative
Original value positive	TP	FN
Original value negative	FP	TN

Where TP, or *true positive*, indicates the number of correct positive imputations (classifications); TN or *true negative* is the number of correct negative imputations; FP or *false positive* is the number of incorrect positive imputations; and FN or *false negative* is the number of incorrect negative imputations.

The three evaluation criteria are:

$$Sensitivity = \frac{TP}{Original\_value\_positive}100\% = \frac{TP}{TP+FN}100\%$$

$$Specificity = \frac{TN}{Original\_value\_negative}100\% = \frac{TN}{FP+TN}100\%$$

$$\Pr edictive\_acuracy = \frac{TP+TN}{total}100\% = \frac{TP+TN}{TP+TN+FP+FN}100\%$$

In other words, the sensitivity measures the percentage of the imputed values, which are positive when the original values were actually positive, i.e. how many of the positive test examples is recognized. The specificity measures the percentage of the imputed values, which are negative when the original values were negative, i.e. how many of the negative test examples is excluded. Predictive accuracy gives the overall evaluation. It is important to notice that a high level of confidence can be placed only for results that yield high values for all three measures: sensitivity, specificity, and predictive accuracy.

2.4.4 Classification

A typical application of the rules generated by ML algorithms is classification. A common feature of ML algorithms is their ability to almost perfectly classify the training set, which corresponds to high correctness of the generated rules. However, the true value of the rules generated by an algorithm should be evaluated only by testing them on new, unseen during learning, data.

Figure 5 shows how an ML algorithm is used to generate and test a data model (rules) using input data, and how the model is used to perform a classification task. First, input data is divided into disjoint training and testing sets. The training set is used to generate rules, while testing data is used to evaluate validity of the generated rules. Once the rules achieve satisfactory quality level, usually in terms of accuracy of describing data from the test set, they are used to perform classification on data that was not used during the training and testing process.

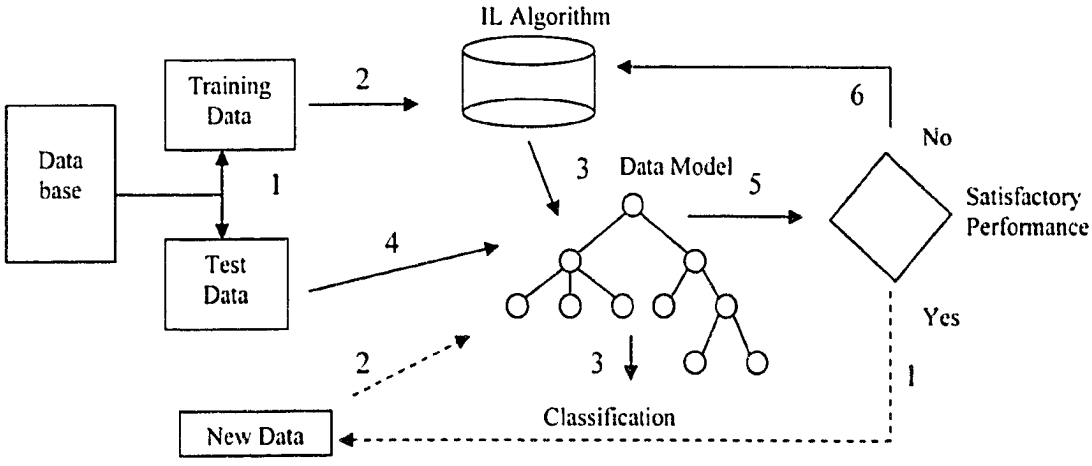


Figure 5. Classification task

The data model generated by ML algorithm can be described in the form of rules or decision trees. A decision tree is capable of expressing knowledge about data described

by a finite number of classes. The tree consists of nodes and labeled edges. Nodes represent attributes, while edges represent possible values of the attributes. The terminal nodes in the tree, called leaves, represent classes. The tree is used to perform a classification of examples by following a path down the tree, starting from the top (root) node, and descending down by following edges, corresponding to the values of the attributes, until a leaf node is reached. The class value assigned to the leaf node defines classification outcome. The decision tree representation is utilized by decision tree algorithms. An example tree is shown in Figure 6.

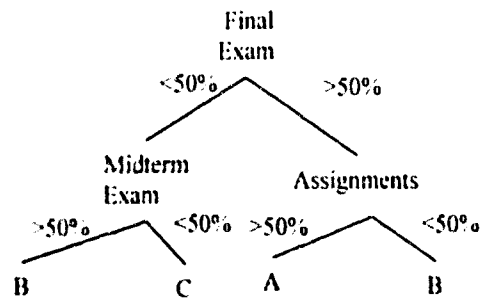


Figure 6. Example of a decision tree

3. Methods for Handling Missing Data

Existing methods for dealing with missing values can be divided into two categories: missing data removal and missing data imputation methods. The removal of missing values is often concerned with discarding the records with missing values, or removing attributes that have missing information. The latter can be applied only when the removed attributes are not needed to perform analysis of the data. Both, removal of records and attributes result in decreasing the information content of the data, and are practical only when a database contains small amounts of missing information, and when an analysis of the remaining complete records will not be biased by the removal [44]. They are usually performed in case of dealing with missing data introduced in the MCAR mode. Another method belonging to this category suggests substituting the missing values for each attribute with an additional category. Although this method provides a simple and easy to implement solution, using it results in substantial problems during the subsequent analysis of the resulting data [73].

3.1 Single and Multiple Imputation Methods

The imputation of missing values uses a number of different algorithms, which can be further subdivided into single and multiple imputation methods. In case of single imputation methods, a missing value is imputed by a single value, while in case of multiple imputations methods, several usually likelihood ordered choices for imputing the missing value are computed [59]. Rubin defines multiple imputations as a process where several complete databases are created by imputing different values to reflect uncertainty about the missing data model, and later each of the databases is analyzed by standard complete-data procedures. At the end, the analyses for each database are

combined into one final result [11] [60]. Detailed description of multiple imputation algorithms can be found in [61] [68], while a primer can be found in [69]. Several approaches have been developed to perform multiple imputations. Li [46], and Rubin and Schafer [58] use Bayesian algorithms that perform imputation using posterior predictive distribution of the missing data based on the complete data. The Rubin-Schafer method assumes the MAR mode, as well as multivariate normal distribution for the data. Azola and Harrell [2] introduce a method that imputes each incomplete variable by cubic spline regression given all other variables, without assuming that the data can be modeled by a multivariate distribution. The commonly used strategies for handling missing values with focus on multiple imputation methods, to impute the missing values in a database related to the airborne particulate matter are reviewed in [39]. In order to evaluate the improvements in accuracy of imputation using multiple imputations, four ad hoc single imputation methods were used. All the ad hoc single imputation methods replace the fully missing values with the sample means of the fully observed data for that attribute, but differ in their imputation of missing values below detection limits. In addition, 95% confidence intervals are considered for the single and multiple imputation methods. From the viewpoint of data analysts building models for multiple imputation, Integrated Moving Average (IMA) seasonal time series model [39] is considered as the most appropriate model for the airborne particulate matter data.

Figure 7 illustrates the flow of operations in a multiple imputation procedure.

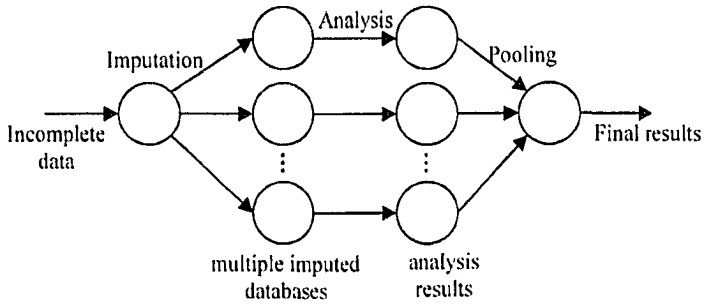


Figure 7. Flow of operations in multiple imputation

Multiple imputation methods are computationally more expensive than single imputation methods, but at the same time better accommodate for sample variability of the imputed value, and uncertainty associated with a particular model used for imputation [44].

3.2 Data Driven, Model Based, and ML Based Imputation Methods

Both, the single and multiple imputation methods can be divided into three categories: data driven, model based, and ML based [44] [47] [52]. Data driven methods use only the complete data to compute imputed values, and assume that the data are fixed, i.e. they are not the outcomes of random variables. Model based methods use data models to compute imputed values, and assume that the data are generated by a model governed by unknown parameters. Finally, ML based methods use the entire available data and a ML algorithm to perform imputation.

3.2.1 Data Driven Imputation Methods

The data driven methods include simple imputation procedures, such as mean, conditional mean, hot deck, cold deck, and substitution imputation [44] [64]. Mean and hot deck methods are described in detail later in the study, while the latter two are only applicable in special cases. The cold deck imputation requires additional database, other

than the database with missing values, to perform imputation, which is usually not available to a data analyst. The substitution method is applicable specifically to survey data, which significantly narrows its application domains.

3.2.2 Model Based Imputation Methods

Several model based imputation algorithms are described in [47]. The leading methods include regression based, likelihood based, and linear discriminant analysis based imputation. In regression based methods, the missing values for a given record are imputed by a regression that uses complete values of attributes for that record. The method requires multiple regression equations, each for a different set of complete attributes, which can lead to high computational cost. Also, different regression models must be used for different types of data, i.e. linear or polynomial models can be used for continuous attributes, while log-linear models are suitable for discrete attributes [44]. The likelihood based methods can be used to impute values only for discrete attributes. They assume that the data is described by a parameterized model, where parameters are estimated by maximum likelihood or maximum a posteriori procedures, which use different variants of the Expectation Maximization (EM) algorithm [15] [47].

3.2.3 Machine Learning Based Methods

Recently, several ML algorithms were applied to design and implement imputation methods. A probabilistic imputation method that uses probability density estimates and Bayesian approach was applied as a preprocessing step for a larger independent component analysis system, which is another example of data analysis tasks that requires complete data [14]. Neural networks were used to implement missing data imputation

methods [36] [73]. We also note the usage of association rule algorithm, which belongs to data mining methods, to perform multiple imputations of discrete data [76]. Recently, supervised ML algorithms were used to design supervised imputation methods. In this case, imputation is performed on one attribute at the time, where the selected attribute is used as a class attribute. A ML algorithm is used to generate a data model from data associated with complete portion of the class attribute, and the generated model is used to perform classification to predict missing values of the class attribute. Several different families of ML algorithms can be used, such as decision trees, probabilistic, and decision rule algorithms [19], while the underlying methodology remains the same. For example, a decision tree algorithm C4.5 [53] [54] and a probabilistic algorithm Autoclass [15] were used in [44], while a decision rule algorithm CLIP4 [16] [17] and a probabilistic algorithm Naïve-Bayes were used in [28]. Another supervised imputation method that combines decision trees and information retrieval principles to develop incremental conditional mean imputation is presented in [20]. Also, a limited comparison between statistical and ML based imputation methods is presented in [29].

The imputation using a ML algorithm is performed by executing multiple classification tasks. Each classification task is performed in two steps. First, during the learning step a ML algorithm generates a model using learning data. The model is used to classify examples into a set of predefined classes, which in case of missing value imputation are all distinct values of an attribute that has missing values. Second, during the testing step, the generated model is used to impute missing data for the testing data, which was not used during learning. The detailed procedure to impute missing values using ML algorithms follows.

First, the attributes that contain missing values are determined. Each such attribute is treated as class attribute in turn, which means that classification task is performed as many times as the number of attributes that contain missing values. Next, the data is divided into training and testing parts. All examples that have a non-missing value in the attribute that is selected as the class attribute are placed in the training set. The remaining examples, i.e. those that have missing information in the class attribute, are placed in the testing set. Next, the ML algorithm is used to generate data model using the training data. The model is applied to the testing data, and classification task is performed to predict values of the class attribute. The predicted values are imputed for the missing values. For each classification, i.e. imputation performed for each attribute that contains missing values, the results in terms of sensitivity, specificity and accuracy of the classification, are recorded. Next, another attribute that contains missing values is selected and the process repeats until all attributes are considered. Finally, the average value of sensitivity, specificity and accuracy across all attributes is computed.

Figure 8 is used to illustrate the above procedure. A database, which contains missing values in all the attributes, is shown in this figure. To impute the missing values, in the first step attribute 1 is set to be the class attribute. Then the database is divided into training set and test set parts. The training dataset is used to develop a data model, which is applied on the testing data to predict the values of the class attribute. In second step attribute 2 is taken as the class attribute and the same procedure as step one is repeated here as well. This procedure continues till imputing the missing values in nth attribute.

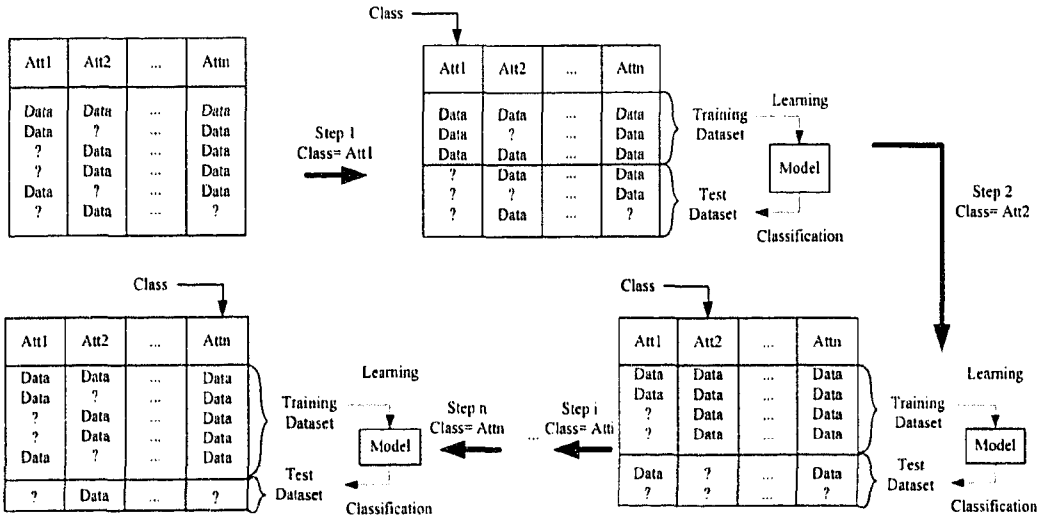


Figure 8. ML imputation process

3.3 History and Motivation

The development of new missing data imputation methods is driven mainly by the need to improve accuracy of imputation. The simplest data driven imputations methods, which were developed first, were followed by model based methods, which in turn were followed by multiple imputation procedures. As a result, very complex algorithms, such as multiple imputation logistic regression methods were developed. Recently, ML based imputations were experimentally compared with data driven imputation, showing their superiority in terms of imputation accuracy [28]. Because of the recent rapid growth of database sizes, the researchers and practitioners require imputation methods that are not only accurate, but also most scalable. Multiple imputations and ML based imputation methods are characterized by relatively high quality, but at the same time are very complex and may be too slow when imputations must be computed rapidly in real time, or for large databases [64].

To this end, a novel framework that aims to improve the accuracy of existing imputation methods while maintaining their computational complexity is proposed in this study. We show, both theoretically and experimentally, that the proposed framework has linear asymptotic complexity, with respect to the number of data points. Therefore as long as the base imputation method has linear or worse complexity (to the best of our knowledge, there are no sub-linear imputation methods), application of the framework does not worsen the base method's complexity. The proposed framework consists of three elements, which are mean pre-imputation, use of confidence intervals and boosting. Extensive experimental tests show that application of the proposed framework improves accuracy of base imputation method and at the same time preserves its asymptotic complexity. The results show that applying the framework to a very simple imputation method, such as hot deck, improves its accuracy of imputation to match accuracy of complex model based imputation methods, such as multiple polytomous logistic regression imputation, while being significantly faster and easier to implement.

This study concerns imputation of discrete attributes. This limitation is imposed by the considered base imputation methods, i.e. in case of considered ML based imputation only discrete attributes can be imputed. We note that the proposed framework is applicable to imputation methods that handle continuous attributes, and its extension to these methods will be the subject of future work.

4. Proposed Framework

All imputation methods used in this study are described in this chapter. These methods include both single and multiple imputation methods. Then the components of the proposed framework i.e. mean pre-imputation module, confidence intervals, and boosting, are explained and analyzed to obtain the asymptotic complexity of the framework.

4.1 Introduction

The proposed framework for improving the accuracy of existing missing data imputation methods is shown in Figure 9. It consists of three main elements, i.e. mean pre-imputation, application of confidence intervals, and boosting, which are shown as gray boxes.

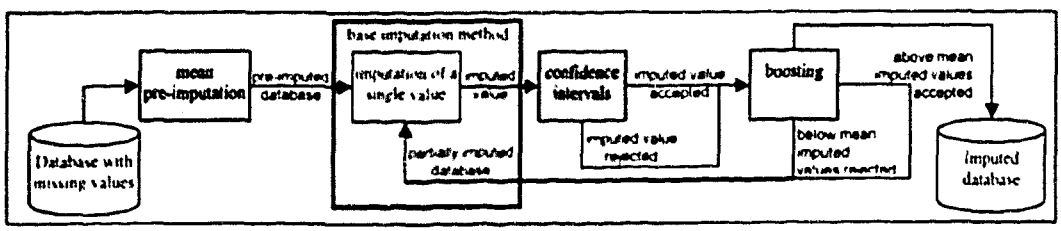


Figure 9.Structure of the proposed framework

The missing values are first pre-imputed, i.e. temporarily filled with a value that is used to perform imputation, using a very fast method such as mean imputation. Next, each missing pre-imputed value is imputed using a base imputation method, and the imputed value is filtered by using confidence intervals. Confidence intervals are used to select the most probable imputed values, while rejecting possible outlier imputations. Once all the values are imputed and filtered, each of them is assigned a goodness value that describes

quality of the imputation, i.e. it usually is expressed as an imputation probability or a distance. Top half of the imputed values are accepted, while bottom half are rejected, and the process repeats with the new partially imputed database. After 10 iterations, all remaining imputed values are accepted, and the imputed database is created. We note that the base methods can be a data driven, model or ML based imputation algorithm. The study tests the framework with two very simple imputation methods, i.e. model driven hot deck imputation and ML based imputation method that uses Naïve Bayes algorithm, to show that these combinations can generate imputations of quality that is higher or the same as the quality of advanced and complex model based methods.

4.2 Relevant Imputation Methods

This chapter provides short description of relevant imputation methods, i.e. methods that were used in the proposed framework, or in the experimental section of this study. A description of how the selected methods are incorporated in the proposed framework is also provided. Selection of the imputation methods was driven by the following assumptions:

- The base methods that will be tested with the proposed framework should be very simple to show that they can be improved by application of the framework to match or surpass quality of complex, high quality model based imputation methods. They should also cover both data driven and ML based categories. Therefore, hot deck imputation and ML based imputation that uses Naïve Bayes algorithms were selected.
- To provide comprehensive evaluation, the framework with the selected two base methods should be compared with advanced, high quality model based imputation methods, as well as fast data driven methods. Therefore, two multiple imputation

methods, i.e. linear discriminant analysis based method and multivariate imputation that combines logistic, polytomous and linear regression, and three data driven methods, i.e. mean, hot deck, and multiple imputation by sampling, are used in the experimental section.

4.2.1 Single Imputation Methods

In the *mean imputation*, mean of values of an attribute that contains missing data is used to fill in the missing values. In case of a categorical attribute, a mode, which is the most frequent value, is used instead of mean. The algorithm imputes missing values for each attribute separately. Mean imputation can be conditional or unconditional. In case of unconditioned mean, the filled mean value is not conditioned on the values of the other variables in the record. Consequently, variance of the imputed variable and its covariance with other variables maybe underestimated. Conditional mean method that imputes the missing values with a mean, which depends on the values of the recorded variables for the incomplete record, was introduced in [8]. In this study, the unconditional mean, which is computationally faster, is used to both impute the missing values as a stand-alone method, as well as a method to perform pre-imputation of the missing values in the proposed framework.

In the *hot deck*, for each record that contains missing values, the most similar record is found, and the missing values are imputed from that record. If the most similar record also contains missing information for the same attributes as the missing information in the original record, then it is discarded and another closest record is found. The procedure is repeated until all the missing values are successfully imputed or entire database is

searched. In case when there is no similar record with the required values filled in, the closest record with the minimum number of missing values is chosen to impute the missing values. There are several ways of finding the most similar record to the record with missing values [61] [63] [29]. In this study, the distance function which is used to measure the similarity between different records assumes distance of 0 between two attributes if both have the same numerical or nominal values, otherwise the distance is 1. The distance of 1 is also assumed for an attribute, for which any of the two records has a missing value. As an example, considering a database described by 4 attributes that has two records with the same value for the first attribute, different values for the rest of the attributes, and a missing value for the fourth attribute in one of the records, the distance between the two records is 2. In the case of supervised databases, it is very important to use classification characteristics of the data because of the correlations (relationships) that exist between the records in the same class. Therefore, to use these characteristics and decrease running time, in this study the distance is computed between the records within the same class.

In *regression*, imputation is performed by regression of the missing values using complete values for a given record [36]. Several regression models can be used, including linear, logistic, polytomous and other. Linear regression applies a linear model, while logistic regression applies maximum likelihood estimation after transforming the missing attribute into a logit variable, which shows changes in natural log odds of the missing attribute. Usually, logistic regression model is applied for binary attributes, polytomous regression for discrete attributes, and linear regression for numerical attributes.

Naïve-Bayes is a ML technique based on computing probabilities [25]. The algorithm works only with discrete data and requires only one pass through the database to generate a classification model, which makes it very efficient, i.e. linear with the number of records. Imputation based on Naïve Bayes algorithm consists of very simple and efficient two steps process. Each attribute is treated as the class attribute, and the data is divided into two parts: training database that includes all records for which class attribute is complete and testing database for which the records are missing. First, prior probability of each non-class attribute value and frequency of each non-class attribute value in combination with each class attribute value is computed based on using the training database. The computed probabilities are used to perform prediction of class attribute values for testing database, which constitute the imputed values.

CLIP4 is a rule-based algorithm that works in three phases [16][17]. During the first phase a decision tree is grown and pruned to divide the data into subsets. During the second phase the set covering method is used to generate production rules. Finally, during the third phase goodness of each of the generated rules is evaluated, and only the best rules are kept while the remaining (weaker) rules are discarded. A specific feature of CLIP4 is use of the integer programming model to perform crucial operations, such as splitting the data into subsets during the first phase, selecting the data subsets that generate the least overlapping and the most general rules, and generating the rules from the data subsets in the second phase. The CLIP4 generates data model that consists of production rules, which use inequalities in all selectors, i.e. IF NUMBER_OF_WHEELS $\neq$ 4 AND ENGINE $\neq$ yes THEN CLASS=bicycle. It works only with discrete data.

C4.5 is a decision tree algorithm [53][54]. It uses an entropy based [70] measure, which is called gain ratio, as a splitting criterion to generate decision trees. Each tree level is generated by dividing the data at a given node into a number of subsets, which are represented by branches. For each division, gain ratio is used to select the best attribute, which values are used to divide the data into subsets. Each subset contains data that takes on one of the values of the selected attribute. C4.5 generates data model that consists of a decision tree, which can be translated into a set of production rules that use equalities in all selectors. It can work with both discrete and continuous data.

4.2.2 Multiple Imputation methods

One of the most flexible and powerful multiple imputation regression based methods is the Multivariate Imputation by Chained Equations (MICE) [9] [10]. The method provides full spectrum of conditional distributions and related regression models. MICE incorporates logistic regression, polytomous regression, linear regression, and uses Gibbs sampler to generate multiple imputation [13]. MICE is furnished with a comprehensive, state-of-the-art missing data imputation software package [40], which is used in the experimental section of this paper, and allows user to specify a different imputation method for each incomplete attribute. It provides Bayesian linear regression for continuous attributes, logistic regression for binary attributes, and polytomous logistic regression for categorical data with more than two categories. MICE also delivers a comprehensive library of non-regression imputation methods, such as predictive mean, unconditional mean, multiple random sample imputation that is suitable for the attributes in the MCAR model, and *linear discriminant analysis* (LDA) for categorical data with more than two categories. LDA is a commonly used technique for data classification and

dimensionality reduction [48], and at the same time serves as a statistical approach to classification based missing data imputation for univariate missing data. LDA method is especially valuable for data where within-class frequencies are unequal, as it maximizes the ratio of between-class variance to the within-class variance to assure best separations.

Table 1 summarizes all methods, which are used in this study. Three single, and four multiple imputation methods were used. The methods include data driven, model based and ML based types. We also note that some of the considered imputation methods work only with discrete attributes. The experimental section used the following imputation methods: random sampling multiple imputation (SAM), mean single imputation (Mean) and hot deck single imputation (HD), regression imputation that uses polytomous and logit multiple imputation (POLYLOGREG), linear discriminant analysis together with logit regression multiple imputation (LDALOGREG), ML based Naïve Bayes single imputation (NB), and Naïve Bayes and hot deck imputations combined with the proposed framework (FNB and FHD, respectively).

Table 1.Summary of the imputation methods used in this study

Method name	Imputation algorithm	Multiple/single imputation	Discrete data	Continuous data	Abbreviation
Naïve-Bayes	Naïve Bayes algorithm	single	Yes	No	NB
Hot deck	nearest neighbor	single	Yes	Yes	HD
Mean	attribute average (mode)	single	Yes	Yes	Mean
Polyreg	polytomous regression	multiple	Yes	No	POLYLOGREG
LDA	linear discriminant analysis	multiple	Yes	No	LDALOGREG
Logreg	logistic regression	multiple	Yes (Binary)	No	---
Sampling	random sampling	multiple	Yes	Yes	SAM
Framework with Naïve-Bayes	Proposed framework with Naïve-Bayes algorithm	Single	Yes	No	FNB
Framework with Hot deck	Proposed framework with Hot deck algorithm	Single	Yes	Yes	FHD

4.3 Detailed Description of the Proposed Framework

Following, each component of the proposed framework, see Figure 9, is explained and its role in the overall framework is described. Also, each component's description is provided and its asymptotic complexity is investigated. We also note that defining n as the number of attributes, r as the number of records, m as max number of missing values for an attribute, and v as the max number of values for an attribute, the following assumptions need to be satisfied: r>>n, r>>v, r>m, and n and v are small constants. Therefore, we assume that complexity is a function of r and n*m, and the remaining variables are omitted.

4.3.1 Mean Pre-Imputation Module

The mean pre-imputation module was developed based on the premise that imputation methods would benefit, i.e. improve their accuracy, by having a complete database to obtain a model and impute the missing data. On the other hand, the pre-imputation should not worsen asymptotic complexity of the entire imputation procedure, and therefore a simple and efficient method should be selected to perform the pre-imputation. Mean imputation was selected as the best candidate for this purpose. Its advantages are simplicity in addition to the acceptable imputation accuracy [28]. Extensive experiments presented in chapter 5 show that mean pre-imputation on average improves imputation accuracy of the subsequently used base imputation method. Also, since computing mode or mean values for each attribute from a given database requires one sweep through the data (for computing mode the attribute values should be encoded into consecutive integers to avoid searching through all attribute values when computing frequencies), the

complexity of performing pre-imputation is linear with respect to the number of records, i.e. O(r), and does not depend on m. This procedure is best described in Figure 10 which presents a pseudo code for the mean pre-imputation procedure. We note that the impact of the mean pre-imputation on the quality of imputation will be shown experimentally later in the study. As it is mentioned before, the data used for this study is either discrete or categorical data. Therefore all the parameters used in the pseudo code should be defined as "Strings" to store the data, which increases the run time of the program and occupies a huge amount of memory that is a big concern in the case of large amount of data. In order to solve this problem, all the data are encoded into the integer values after loading them from the database and is used in all parts of the program. This encoding makes it possible to save the data in integer tables. Consequently the run time of the program decreases and memory is used more efficiently. In this study the pseudo code related to the components of the framework are described to find the asymptotic complexity of the proposed framework. After finishing the imputation procedure, the encoded values will be decoded to their original values.

Figure 10 shows the pseudo code for the mean pre-imputation procedure.

The following is the description of the terms used in the Figure 10:

List: is a r*n table that stores the whole encoded data

Position: is a table that keeps the location of the missing values related to the whole database so its dimension is a function of the position and amounts of missing values.

mode: represents the mode (most frequent value) of the values in each attribute

mode-count: is a one dimensional array to count the frequency of each value in one attribute

mode1: represents the frequency of the most frequent value

```
Given: Position, List, n (number of attributes),
1    Initialize mode = 0,mode1 = 0;
2    for j = 1 to n                                          //for all attributes
2.1      if ( Position[j][0] ≠ 0)
2.1.1        then for i = 1 to v
2.1.1.1                  initialize mode-count[] = 0,
2.1.2            for i = 1 to r                              //for all the records
2.1.2.1                  if (List[i] ≠ "?") then  mode-count[List[i]]++;   // ? stands for the missing value
2.1.3            for i = 1 to numberofvalues                 //for all values in each attribute
2.1.3.1                  if( mode-count[i] > mode1)
2.1.3.2                    then mode1 = mode-count[i]; mode = i;
2.1.4            for i = 1 to Position[j][0]
2.1.4.1                  List[j][Position[j][i+1]] = mode;
```

Figure 10. Pseudo code of the mean pre-imputation procedure

At the end all the missing values in each attribute in the **List** table are replaced by the mode of that attribute.

Also several "for loops" exist in the pseudo code that are described as following:

The first "for loop" is repeated n times and each time one attribute is taken to impute its missing values with the mode (most frequent value) of the attribute.

Loop in line 2.1.1: is to initialize the mode-count with zero

Loop in line 2.1.2: is to count the frequency of each value in the attribute

Loop in line 2.1.3: finds the most frequent value in the attribute

Loop in line 2.1.4: fills all the missing values in the list table with the mode (most frequent value)

In the following, the asymptotic complexity of the mean pre-imputation method is determined. The size of the Position array is at most nO(m) just because some attributes may not contain missing values, List array is nO(r), Type is n and mode-count is O(v). To estimate the complexity of the mean pre-imputation process, the complexity of each step, is calculated separately as follows:

Asymptotic complexity of the step 1, i.e. initialization is:

nO(m) to derive Position matrix

nO(r) to derive List matrix

nO(1) to derive Type matrix

Thus, the total complexity of step 1 is nO(m) + nO(r) + nO(1) = nO(r).

Asymptotic complexity of step 2 is:

The entire step is repeated n times, and therefore complexity of steps 2.1 through 2.1.4.1 will be multiplied by n.

Line 2.1	O(1)
Line 2.1.1	O(v) and applies to line 2.1.1.1 nesting in pseudo-code
Line 2.1.1.1	O(1)
Line 2.1.2	O(r) and applies to line 2.1.2.1
Line 2.1.2.1	O(1)
Line 2.1.3	O(v) and applies to lines 2.1.3.1-2
Line 2.1.3.1	O(1)
Line 2.1.3.2	O(1)
Line 2.1.4	O(m) and applies to line 2.1.4.1
Line 2.1.4.1	O(1)

The complexity of step 2 is: O(n) * [O(1) + O(v)*O(1) + O(r)*O(1) + O(v)*(O(1) + O(1)) + O(m)*O(1)] = O(nr). The overall complexity is nO(r) + O(nr) = O(nr) = O(r).

So overall asymptotic complexity of the mean pre-imputation is a linear function of the size of database.

4.3.2 Confidence Intervals Module

Confidence intervals for the mean are an interval estimate for the mean. Interval estimates are often desirable because the estimate of the mean varies from sample to sample. Instead of a single estimate for the mean, a confidence interval generates a lower and upper limit for the mean. The interval estimate gives an indication of how much uncertainty there is in our estimate of the true mean. The narrower the interval, the more precise is our estimate. Confidence limits are expressed in terms of a confidence coefficient. As a technical note, a 95% confidence interval does not mean that there is a 95% probability that the interval contains the true mean. The interval computed from a given sample either contains the true mean or it does not. Instead, the level of confidence is associated with the method of calculating the interval. The confidence coefficient is simply the proportion of samples of a given size that may be expected to contain the true mean. That is, for a 95% confidence interval, if many samples are collected and the confidence interval computed, in the long run about 95% of these intervals would contain the true mean.

In this study the confidence intervals module is used to filter out possible outlier imputation candidates that are generated by the base imputation method. The filter is

based on the premise that imputed values, which are close to the mean (for numerical attributes) or mode (for nominal attributes) of an attribute, have the highest probability of being correct. The filter is designed by computing so called confidence intervals. Imputed values for a given attribute that are within the intervals are kept, while the values outside of the interval are discarded. The confidence intervals are defined as an interval estimate for the mean of an attribute [72]. Confidence intervals define a lower and upper limit for the mean, which are defined as $M - z\sigma_M \leq X \leq M + z\sigma_M$, where M is the sample mean, $\sigma_M = \frac{\sigma}{\sqrt{r}}$ is standard error of the mean, σ is the standard deviation of the original distribution, r is the number of records, and the value of z depends on the desired level of confidence. This definition applies to numerical attributes.

In the case of nominal attributes, mean is substituted by mode, and frequency of values for an attribute is computed and normalized as follows: value of 1 is assigned to the most frequent value for the attribute, 0 is assigned to the frequency of zero, and the frequencies of the remaining attribute values are assigned a normalized value within [0,1]. By analogy to the confidence intervals for numerical attributes, the confidence intervals for nominal attributes are defined as: $f_{avg} \leq X \leq 1$, where f_{avg} is an average value of the normalized frequencies for all attribute's values. In other words, imputed values with a frequency lower than the average will be filtered out. To further improve quality of the filter, for all supervised databases the confidence intervals are computed individually for each of the predefined classes, i.e. a confidence interval is computed for each subset of the database that is associated with a given class value. When applying the computed intervals to filter an imputed value, an interval is used that corresponding to the class

value of the records for which the value is imputed. We note that normalizing and computing average frequencies for all values for each attribute from a given database requires one sweep through the data (again, the attribute values should be encoded), the complexity of computing confidence intervals is linear with respect to the number of records, i.e. O(r), and does not depend on m. Also, filtering the imputed values using the confidence intervals requires O(n*m) time since filtering each missing value takes O(1) time, and therefore the complexity of the confidence intervals module is O(r) + O(n*m), and is linear with respect to the number of records and the total number of missing values. Computation and application of confidence intervals for numerical attributes have also linear complexity. The details of the required procedure are shown in Figure 11, which represents the pseudo code for computing the confidence intervals.

```
Given array[]
1       Initialize Numbers[] = 0, Count = 0; Interval[] = 0;
2.1     for j = 1 to r                                          //for all records
2.1.1     if (array[j] ≠ "?")
2.1.2       then Numbers[array[j]]++;
2.2     max = Numbers[0]; min = Numbers[0]; Tnumbers = 0;
2.3     for k=1 to v
2.3.1     if (max < Numbers[k])
2.3.2       then max = Numbers[k];
2.3.3     if (min > Numbers[k])
2.3.4       then min=Numbers[k];
2.4     for k=1 to v
2.4.1     Numbers[k] = (Numbers[k] / max);                      //normalizing the frequencies
2.4.2     Tnumbers+ = Numbers[k];
2.5     TNumbers = Tnumbers / v;
2.6     for k = 1 to v
2.6.1     if (Numbers[k] > Tnumbers or min = max)
2.6.2       then Interval[Count] = k; Count++;
```

Figure 11. Pseudo code for the procedure of computing the confidence intervals

We note that the impact of the confidence intervals module on the quality of imputation will be shown experimentally later in the study.

In the above pseudo code the following terms are used:

Array: represents each attribute of the database.

Numbers: represents the frequency of each value in the attribute.

Count: counts the number of values existing in the interval.

Interval: is a set of values that constitute the intervals.

Tnumber: is the total number of records in the attribute ignoring the missing values.

Description of the loops existing in the pseudo code is as following:

Loop in line 2.1: obtains the frequency of each value in the current attribute.

Loop in line 2.3: finds the most frequent and less frequent values in the database.

Loop in line 2.4: normalizes the frequencies with respect to the most frequent value, therefore the most frequent value will have a normalized frequency equal to 1.

Loop in line 2.6: selects all the values that have a frequency higher than the average and stores the data in an array called Interval.

Following, asymptotic complexity of the procedure for computing the confidence intervals is determined. The complexity of each step, is calculated separately as:

Complexity of the step 1, i.e. initialization:

O(r) to derive array matrix

Thus, the total complexity of step 1 is O(r).

Complexity of step 2 is:

The entire step is repeated n*c times, where c is the number of classes in a supervised database (c=1 for unsupervised database), and therefore complexity of steps 1 through 2.6.2 will be multiplied by n*c.

Line 1	O(v) + O(1) + O(v) = O(v)
Line 2.1	O(r) and applies to lines 2.1.1 and 2.1.2
Line 2.1.1	O(1)
Line 2.1.2	O(1)
Line 2.2	O(1)
Line 2.3	O(v) and applies to lines 2.3.1through 2.3.4
Line 2.3.1	O(1)
Line 2.3.2	O(1)
Line 2.3.3	O(1)
Line 2.3.4	O(1)
Line 2.4	O(v) and applies to lines 2.4.1 through 2.4.2
Line 2.4.1	O(1)
Line 2.4.2	O(1)
Line 2.5	O(1)
Line 2.6	O(v) and applies to lines 2.6.1 through 2.6.2
Line 2.6.1	O(1)
Line 2.6.2	O(1)

The overall complexity is O(nc) * [O(v) + O(r)*(O(1) + O(1) + O(1)) + O(v)*(O(1) + O(1) + O(1) + O(1)) + O(v)*(O(1) + O(1) + O(1)) + O(v)*(O(1) + O(1))] = O(ncr) = O(r).

Thus, the overall complexity of the confidence intervals procedure is a linear function of the size of database. Also, filtering the imputed values using the confidence intervals requires O(1) time, and therefore the complexity of the confidence intervals module is O(r). Computation and application of confidence intervals for numerical attributes have also linear complexity.

4.3.3 Boosting

Boosting is a ML procedure for improving the accuracy of classification algorithms [33] [65]; full reference list can be found at http://www.boosting.org/. The underlying idea of boosting is to combine simple "rules" to form an ensemble such that the performance of the single ensemble member is improved. Imagine $h_1, h_2, ..., h_t$ are a set of parameters, and consider the function

$$f(x) = \sum_{i=1}^{T} c_i h_i(x).$$

Here c_i denotes the coefficient with which the ensemble member h_i is combined; both c_i and the learner or parameter h_i are to be learned within the boosting procedure.

Keans and Valiant [41] proved that learners, each performing only slightly better than random, can be combined to form an arbitrary good ensemble hypothesis when enough data is available. Schapire [67] was the first one who provided a provably polynomial time boosting algorithm and the *AdaBoost* (**Ada**ptive **Boost**ing) algorithm [32][34] is generally considered as a first step towards more practical boosting algorithms. *Arcing* algorithm is a similar method to AdaBoost, which converges to a linear programming solution [7]. An interesting step towards practical applications states that large parts of

the early boosting literature persistently contained the misconception that boosting would not over fit even when running for a large iterations. Simulations by [37][57] on datasets with higher noise content show the overfitting effects, which can be avoided by regularizing boosting so as to limit the complexity of the function class. It is important to clarify the relations between optimization theory and boosting procedures in order to develop means for achieving robust boosting [7][31][35]. Developing this relationship can introduce new types of boosting algorithms. Some examples are boosting algorithms for regression [26][55], multi-class problems [1][56], unsupervised learning [12][56] and to establish convergence proofs for boosting algorithms by using results from the theory of optimization. Further extensions to boosting algorithms can be found in [12][56][35][67][71][1][26][49][23]. Recently, boosting strategies have been quite successfully used in various real-world applications. For example in [22] boosting was used for tumor classification with gene expression data. For further applications and more details we refer to the www.boosting.org/applications.

Boosting in its original version is a procedure where a set of data models is iteratively generated by an algorithm for a given dataset based on modification of weights associated with records. The weights are modified to increase focus of the next model on generating correct model for records that were misclassified by the preceding classifiers. The classification generated by individual models is combined using a voting schema to generate classification outcome. In general, both theoretical and experimental studies show that boosting which is a weak classification algorithm, i.e. algorithms that generated models better than random drawing, results in a classification that is more accurate than a model generated by a strong classification algorithm.

A boosting-like technique was applied in the proposed framework. The main goal was to improve the accuracy of the imputation by accepting only high quality imputed values and using them, i.e. additional and reliable information, to impute the remaining values. The module is appended at the end of the imputation process, when all imputed values are already filtered. It works iteratively, where in each iteration it selects and accepts top, high quality imputed values, while rejecting remaining values. In this way, a partially imputed database is created and fed back to the base imputation algorithm. Next, the process repeats, but this time concentrating on imputing the remaining values. The number of iterations is set to 10, where in the last iteration all remaining imputed values are accepted. The imputed values are accepted or rejected based on their weight and a threshold, i.e. all values with weights above the threshold are accepted while remaining values are rejected. The weights should represent quality of imputation, and are dependent on a particular base imputation method. In this study two base imputation methods, i.e. Naïve-Bayes ML imputation method and hot deck imputation method, are investigated. In case of Naïve-Bayes ML imputation, the weights are defined as the probability of the top class variable, i.e. probability that is used to perform imputation. The threshold is set to be the mean top class probability for all imputed values. Similarly, for the hot deck imputation the weights are defined as the distance between the record with imputed values and the records from which the imputed value was taken, i.e. the two records that are used to perform imputation. The threshold is set as the average distance between the records with missing data and their closest records for all the imputed values. Since weight values are taken directly and without additional computational cost from the base imputation methods, computation of the threshold requires O(n*m) time since it

involves computing mean value among weights for all imputed values, the selection/rejection of imputed values takes O(n*m) time since filtering each imputed value takes O(1) time, and therefore the complexity of the boosting module is O(n*m). The complexity is constant with respect to the number of records, and linear with respect to the total number of missing values. The pseudo code related to the boosting procedure, which is applied to the Naïve-Bayes imputation method is shown in Figure 12. We note that impact of the boosting on the quality of imputation will be shown experimentally later in the study.

```
1.1     for i=1 to numberofvalues                  // adding up all the probabilities
1.2         Mean-Prob = prob[i];                   // find the average probability
2.1     Mean-Prob = Mean - prob nofmissing;
2.2     for k=1 to m                               // for all the missing values in that attribute
2.2.1       if (prob[k] > Mean-Prob)
2.2.2       then List[k] = final-decision[k];
```

Figure 12. Pseudo code of the boosting procedure for the Naïve Bayes imputation method as the base method

The boosting part of the framework is coupled with the base imputation method. As an example the pseudo code related to boosting the Naïve-Bayes algorithm is shown in Figure 12. After calculating the probability of each imputation candidate, the result is added to the previous probabilities (line 1.1 and 1.2) in order to find the average probability of the candidates in line 2.1.

The loop in line 2.2 and the "if condition" in line 2.2.1 only accept the imputed values which have a probability higher than the average and locate them into the database (List) instead of the missing values. The number of missing values, *m*, can be different in each attribute so based on the attribute which is in process, the loop will be repeated *m* times.

The values which do not satisfy the condition are imputed in the next iterations and this process is repeated 10 times. All the imputed values are accepted during the last iteration.

Complexity of the method

Line 1.1	O(v)	// and applies to 1.2
Line 1.2	O(1)	
Line 2.1	O(1)	
Line 2.2	O(m)	// and applies to 2.2.1-2.2.2
Line 2.2.1	O(1)	
Line 2.2.2	O(1)	

The total complexity of the confidence interval module is:

Number-Of-Iterations*[O(v)*O(1)+O(1)+O(m)*[O(1)+O(1)]]=Number-Of-Iterations*O(m)=O(m).

This process will repeat for each attribute so in general the total complexity is O(n*m).

The overall complexity of the proposed framework is computed as the complexity of pre-imputation, i.e. O(r), summed with complexity of the confidence intervals module and boosting module multiplied by number of boosting iterations, i.e. 10*(O(r) + O(n*m) + O(n*m)). Therefore the total complexity is O(r) + 10*(O(r) + O(n*m) + O(n*m)) = O(r) + O(n*m), and is linear with respect to both the number of records and the total number of missing values.

5. Experiments and Results

This chapter, which is composed of two parts, experimentally investigates the effect of the proposed framework on improving the performance of the imputation methods. In the first, several data imputation methods from different groups are experimentally compared to evaluate their performance. Then in the second part, the proposed framework is applied to two imputation methods, i.e. Hot deck and Naïve-Bayes, and the effect of each component on improving the imputation accuracy is investigated. Then the results of imputation are compared with other state of the art methods.

5.1 Experimental Analysis of Imputation Methods

The first set of experiments concentrate on performing experimental analysis of several algorithms for imputation of missing values, which range from simple statistical algorithms like mean and hot deck imputation to imputation algorithms that work based on application of inductive ML algorithms. Three major families of ML algorithms, such as probabilistic algorithms (e.g. Naïve Bayes), decision tree algorithms (e.g. C4.5), and decision rule algorithms (e.g. CLIP4), are used to implement the ML based imputation algorithms. The experiments were performed using seven different datasets, and the five missing data imputation algorithms. The selected seven datasets originally do not contain missing values. The missing data were introduced artificially, using the MAR model, into each of the datasets. As a result missing values were introduced into all attributes, including class attribute. The missing data was artificially generated to enable verification of the quality of imputation, which was preformed by comparing the imputed values with the original values.

In what follows, first the seven selected datasets are introduced and described. Each dataset is described by a set of characteristics. The selected datasets cover entire spectrum of values for each of the characteristics. Next, the imputation experiments are described and explained. Finally, the results of experiments are investigated to analyze possible links between the characteristics of input datasets and quality of imputation of specific algorithms.

5.1.1 Experimental Setup

The experiments use seven datasets selected from the UCI ML repository [4]. The selected datasets include only discrete attributes, since both Naïve Bayes and CLIP4 ML algorithms, which are used to perform supervised imputation, cannot work with continuous attributes. The description of the selected datasets, ordered by the number of examples, is shown in Table 2. It should be noted that only datasets which are specified by gray color, both dark and light, are used in this chapter that are Len, Hay, Tic, Car, Krs, LED and Nrs datasets. The following characteristics are considered for this experiment.

- the size of datasets, expressed in terms of the number of examples ranges between 24 and almost 13K
- the number of attributes ranges between 4 and 36
- the number of classes ranges between 2 and 10
- the ratio of Boolean attributes ranges between 0 and 97%

In general, the datasets were selected to assure comprehensiveness of the results. The experiments introduce missing values in four different quantities, i.e. 5%, 10%, 20% and

50% of data was randomly turned into missing values. This assures that entire spectrum, in terms of amount of missing values, is covered.

The quality of imputation was evaluated by comparison of imputed values with the original values. The experiments report accuracy of the imputation. For the supervised imputation methods, sensitivity and specificity of the imputation are also computed. These values are computed for each of the attributes in the data, and the average value is reported.

5.1.2 Relevant Database Characteristics

Based on the experimental results, several changes were made in respect to the choice and design of the database characteristics initially considered and described in part1. We note that these characteristics were designed for general data analysis purposes, not just for the missing data imputation task. While analysis of results with respect to some characteristics, such as number of attributes and number of examples, generated some interesting knowledge, analysis for the remaining characteristics, i.e. number of classes and proportion of Boolean attributes, did not generate useful knowledge showing that their definitions need to be redesigned.

In general, ML algorithms depend not only on the number of classes, but more properly on the number of examples for each class. Therefore, in this study, "number of examples/ number of classes" characteristic is used instead of the "number of classes" characteristic. Similar reasoning applies to the "proportion of Boolean attributes" characteristic. Using a simple proportion does not accommodate for the characteristics of the remaining, non Boolean, portion of the data, which is important from the ML point of view. We note that ML algorithms can be sensitive to granularity of attributes expressed in terms of number

of their distinct values combined with the number of classes defined in the data. For example, attributes with number of distinct values lower than number of classes cannot be successfully used to distinguish between all classes. This lead to defining a new characteristic "number of Boolean values / (number of values*number of classes)", which was used instead of the "proportion of Boolean attributes" characteristic. Also, a new "amount of missing values" characteristic was added. Therefore, the following new characteristics are used to describe the input databases in order to come up with guidelines to select the most suitable missing data imputation methods:

- Amount of missing values,
- Number of examples,
- Number of attributes,
- Number of Boolean values/(number of values*number of classes),
- Number of examples/number of classes.

Next chapter provides and analyzes comparison of imputation methods based on the above characteristics.

5.1.3 Comparison of the data imputation methods

The results section summarizes experiments that apply five missing values imputation methods on seven datasets, for which four different amounts of missing information were introduced. The results report accuracy of the imputation, and are analyzed from the perspective of each of the input data characteristics.

First, Figure 13 summarizes imputation accuracy of each method against the four amounts of missing values. The accuracies for each amount of the introduced missing values are averaged over the seven datasets. Figure 13 shows that the supervised imputation method based on the C4.5 ML algorithm has on average the best imputation accuracy throughout the entire considered spectrum of amounts of missing values. The supervised imputation method based on the Naïve Bayes ML algorithm has the mean imputation accuracy, which is very close to the accuracy of the imputation based on the C4.5 algorithm. The Mean imputation method has, on average, the mean imputation accuracy that places it on the third position, while the remaining methods are significantly worse. In general, we observe that the supervised imputation algorithms have better performance comparing to the unsupervised algorithms. Among the supervised algorithms, method based on the C4.5 ML algorithm, which is a decision tree algorithm, has the best mean imputation accuracy across the different amounts of missing values. Figure 2 shows that, in general, the imputation accuracy of all imputation methods declines with the increasing amount of missing information, which is a result of poorer quality of the underlying data.

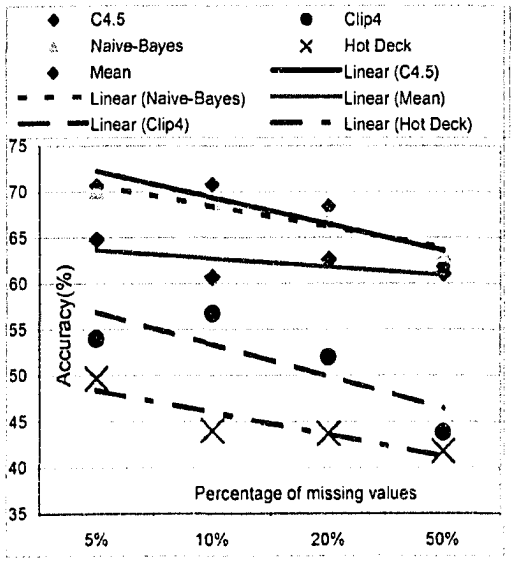


Figure 13. Accuracy against amount of missing values

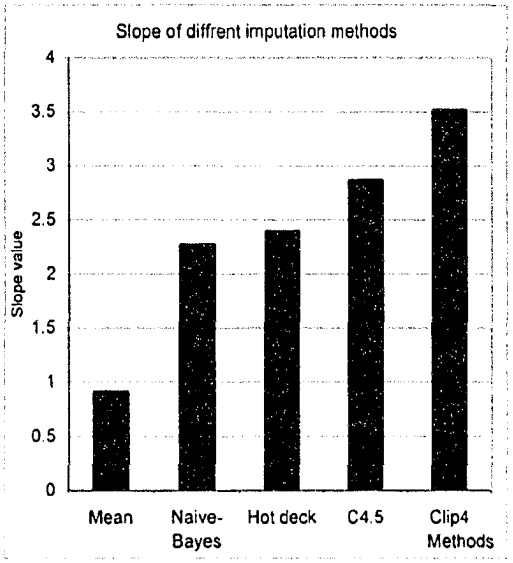


Figure 14. Slope of the accuracy trend for different amounts of missing values

Figure 14 shows the slope of the linear trend between the mean imputation accuracy and the different amounts of missing information. The slope shows the pace of performance degradation of each of the missing data imputation methods with the increasing amount of missing data. The higher the value of the slope the faster the quality of the method degrades. It can be observed that the Mean imputation method is the most stable imputation method. Although it has lower mean imputation accuracy for the considered amounts of missing information than the supervised methods based on the C4.5 and Naïve Bayes methods, its stability suggests that for higher amounts of missing values it may overrun the supervised imputation methods. We note that in general datasets contain small amount of missing information, but for some domains it is possible to have more then 50% of missing values. For example, a medical data describing patients with cystic fibrosis that contains over 65% of missing information was successfully used to find useful relationships about the disease [43].

To summarize, Figure 14 shows that the unsupervised imputation methods are more stable comparing to the supervised methods. The main reason is that the supervised methods must have a training dataset of proper quality to develop an accurate model that is used to impute the missing information. On the other hand, the unsupervised imputation methods are less sensitive to the amount of missing values.

Figure 15, Figure 16, Figure 17, and Figure 18 compare different missing data imputation methods based on the normalized rank values. The normalized rank enables side by side comparison of the imputation methods, which is independent of the quality of the considered datasets. In order to compute the normalized rank value, the imputation accuracy of all methods is scaled to a common [0, 1] interval, with the lowest accuracy set to 0, and highest accuracy set to 1. The remaining imputation accuracy values are computed proportionally within the interval. For example, if the lowest accuracy for a given method would be 60% and the highest 90%, then 90% becomes 1, 60% becomes 0, and the scaled value for 80% accuracy would be 0.667.

Figure 15 shows the normalized rank values for the average imputation accuracy, across different amounts of missing values, for all imputation methods against the increasing number of examples in the datasets. The rank for both CLIP4 and Naïve Bayes based supervised imputation methods improves with the increasing size of the dataset. We note that in general the amount of input data is an important factor for ML algorithms. Having more data may help the ML algorithms to generate a better model, which consequently improves the quality of imputation. We also note that the supervised imputation method based on the C4.5 ML algorithm almost always performs the best. The quality of the imputation performed with the unsupervised imputation methods does not dependent on

the size of the data. There is no clear trend in their performance for the increasing amount of input data.

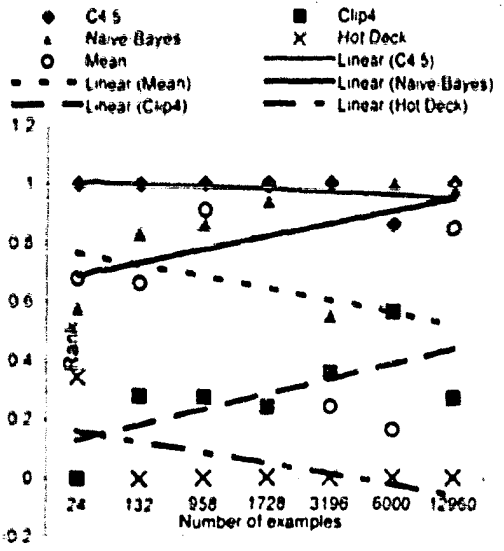


Figure 15. Normalized rank of the average imputation accuracy versus the number of examples

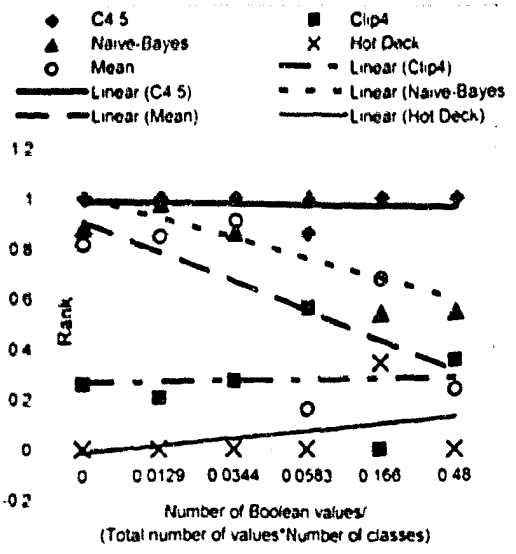


Figure 16. Normalized rank of the average imputation accuracy versus the Boolean characteristic

Figure 16 shows the normalized rank values for the average imputation accuracy, across different amounts of missing values, for all imputation methods against the *Boolean characteristic*, which is defined as "number of Boolean values / (total number of values*number of classes)". We observe that for the increasing values of this characteristic, performance of the supervised imputation method based on the Naïve Bayes algorithm gets worse comparing to the method based on the C4.5 ML algorithm. The same trend can be observed for the Mean imputation method. Other methods are not susceptible to this characteristic, and the imputation method based on the C4.5 ML algorithm has the best performance.

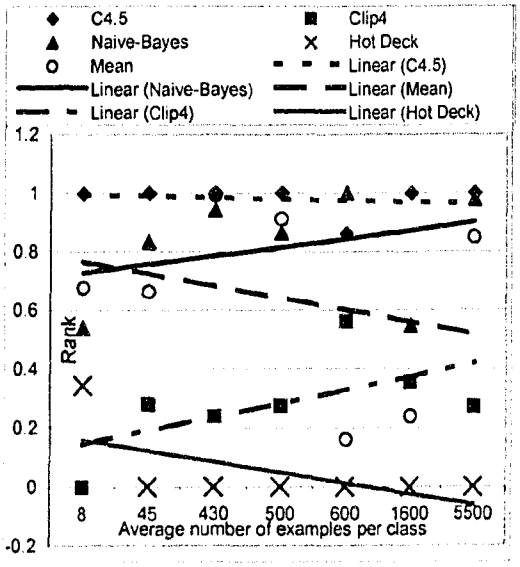


Figure 17. Normalized rank of the average imputation accuracy vs. the average number of examples / class

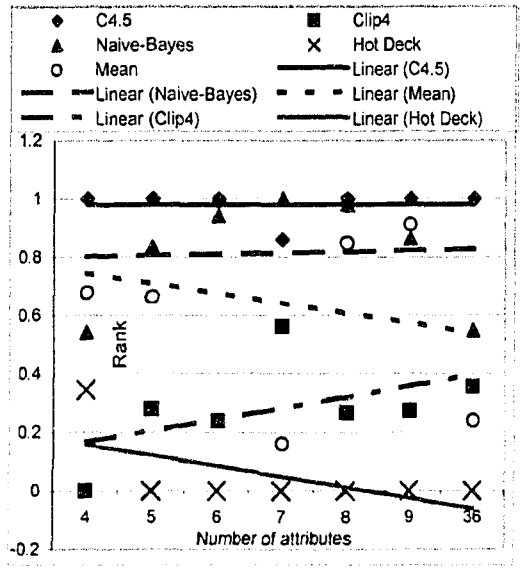


Figure 18. Normalized rank of the average imputation accuracy versus the number of attributes

Figure 17 shows the normalized rank values for the average imputation accuracy, across different amounts of missing values, for all imputation methods against the average number of examples per class, which is defined as "number of examples / number of classes". The Figure shows that the quality of imputation for the supervised imputation methods based on the CLIP4 and Naïve Bayes algorithms improves with the increasing value of the average number of examples per class. This can be attributed to the improved quality of the data model generated by the ML algorithms with the increasing number of examples that are used to generate it. As a result, the quality of the imputation that is performed using the generated data model improves. On the other hand, it can be observed, as expected, that the unsupervised imputation algorithms are not susceptible to this characteristic. We also note that the imputation method based on the C4.5 algorithm has the best average normalized rank.

Figure 18 shows the normalized rank values for the average imputation accuracy, across different amounts of missing values, for all imputation methods against the number of attributes. The rank of supervised imputation methods based on both CLIP4 and Naïve Bayes algorithms improves with the increasing number of attributes. This trend can be attributed to the increasing quality of the data models used to perform imputation, similarly as for the results described in the Figure 17. Again, we note that imputation method based on the C4.5 algorithm has the best average normalized rank.

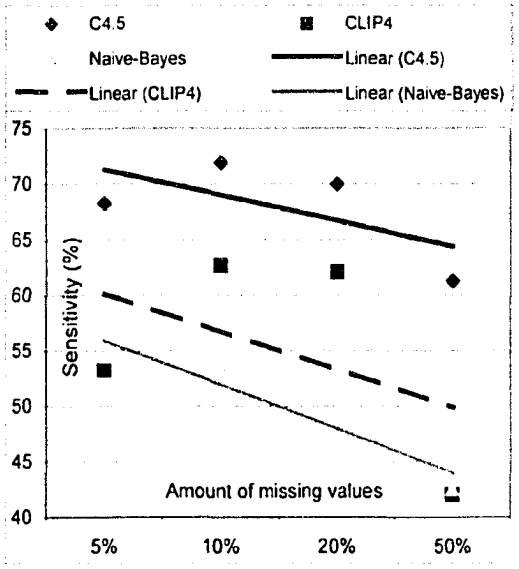


Figure 19. Average sensitivity versus the amount of missing values

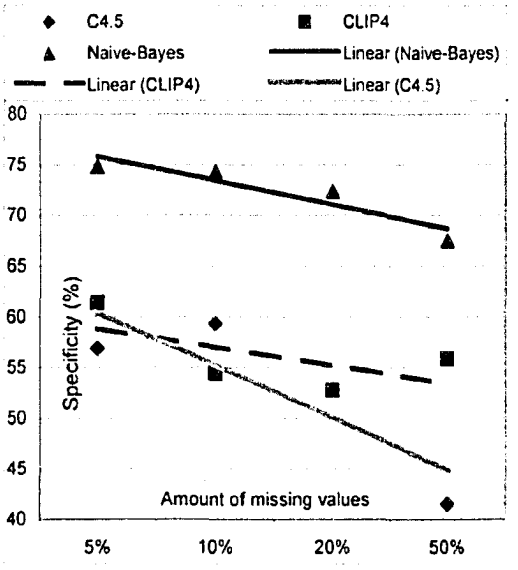


Figure 20. Average specificity versus the amount of missing values

The average sensitivity, over the seven input datasets, of the supervised imputation methods against the different amounts of the missing data is shown in Figure 19. Similarly, Figure 20 shows the average specificity of different imputation methods. In general, increasing the amount of missing values results in decline of both sensitivity and

specificity for all supervised imputation methods. In case of the sensitivity, the slope of the trend line for the imputation method based on the Naïve-Bayes and CLIP4 ML algorithms is greater than the method based on the C4.5 algorithm. We also note that the imputation method based on the C4.5 algorithm achieves the best sensitivity values. On the other hand, the specificity of this method is worse than specificity of the method based on the Naïve Bayes algorithm. This shows that the imputation method based on the C4.5 is not universally better than imputation based on the Naïve Bayes algorithm, but rather they complement each other.

5.1.4 Discussion

The results shown in Figure 13 through Figure 18 indicate that the supervised imputation method based on the C4.5 ML algorithm has the best overall performance. The results also indicate that the imputation method based on the Naïve Bayes ML algorithm is the second best. In general, it can be seen that the supervised imputation methods have better performance than the unsupervised imputation methods.

The analysis of stability of performance of the imputation methods with the increasing amount of missing values shows some interesting relationships. The Mean imputation is the most stable, which means that its performance degradation is the slowest compared to all other methods considered in this study. We expect that for the datasets with high amounts of missing values, unsupervised imputation algorithms may perform better than the supervised one. The rationale behind it is that supervised methods build data model which is used to perform imputation and which quality is dependent on the quality of underlying data, while unsupervised methods are more robust in terms of the quality of the underlying data.

Another important trend shows that increasing the number of attributes and number of examples results in increasing the quality of imputation for the supervised imputation methods. Comparison between the sensitivity and specificity of different supervised imputation methods shows that although the C4.5 based method has better sensitivity, the Naïve Bayes based method is superior in terms of specificity. This shows that these methods complement each other.

The results also show that the performance of the unsupervised imputation methods does not depend on the number of attributes, which conforms to the procedures they use.

We also note that although the execution time of the imputation algorithms was not measured, in general the unsupervised Mean imputation method was the fastest and scaled well with the increasing size of the input data. The second fastest was the supervised imputation that uses the C4.5 algorithm.

5.2 Missing Data Imputation Using the Proposed Framework

The proposed framework was tested with a wide range of databases to experimentally verify its advantages.

The experiments were performed using fifteen real databases. These databases are mostly supervised and chosen from UCI ML repository [4] and KDD repository [38]. The selected databases initially do not contain missing values. The missing data were introduced artificially, using the MCAR mechanism, into each of the databases. As a

result missing values were introduced into all attributes, except the class attribute. Hence the classification characteristic of the supervised databases can be used to improve the accuracy of imputations. The missing data was artificially generated to enable verification of the quality of imputation, which was performed by comparing the imputed values with the original values. The introduced missing values are in six different quantities, i.e. 5%, 10%, 20%, 30%, 40% and 50% to cover the entire spectrum of amount of missing values.

In the following, first the fifteen selected databases are introduced and described. Each database is described by a set of characteristics. The selected databases cover the entire spectrum of values for each of the characteristics. Next, the effect of each component in the framework on the improvement in accuracy of imputation is experimentally demonstrated. Also the results of experiments obtained from the framework with Naïve-Bayes and Hot deck imputation methods are compared to the results from other imputation methods. In addition, this chapter considers the run time of the all imputation methods as well as the time scalability analysis related to the asymptotic complexity of the proposed framework.

The imputation procedure is depicted in Figure 21. As it is shown on the figure, the database is initially a complete database, which makes it possible to introduce different amounts of missing values in it. In this figure the missing values are shown by "?". In the next step the missing values are imputed using the imputation methods. Therefore all the "?" are replaced by "IData" which represents the imputed data. Finally the results of imputation are compared with the original values to calculate the accuracy of imputation.

The number of correct imputations over the total number of missing values determines the accuracy of the imputation method.

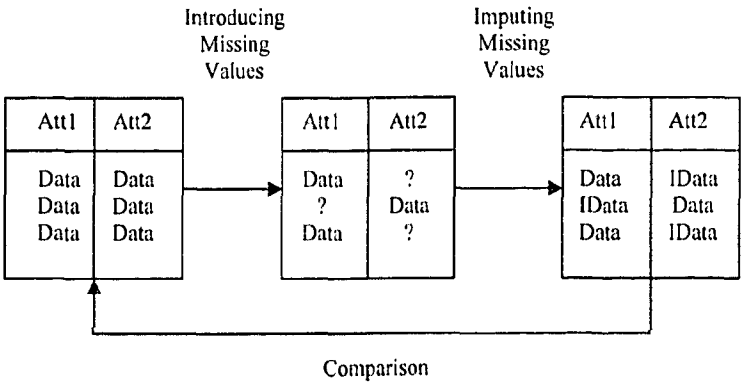


Figure 21. Imputation procedure including introducing missing values and imputing the missing values

5.2.1 Experimental Setup

The experiments use sixteen databases from the UCI ML and KDD repository. The selected databases include only discrete attributes, since Naïve-Bayes ML algorithm, which is used to perform supervised imputation, cannot work with continuous attributes. However the proposed framework can be applied to the methods that deal with both continuous and discrete values such as Hot deck imputation. The description of the selected databases, ordered by the number of examples, is shown in Table 2. Only datasets in white color or light gray color rows are used in this part of the experiments.

The *syn* is a synthetic dataset being generated using a dataset generator published at www.datasetgenerator.com, and is used to evaluate complexity of the considered methods. The dataset was generated using the following settings: number of predicting attributes was set up as 20, domain size of the attributes is equal to 20, number of rules is 10, and number of records was taken as 256,000.

Table 2.Description of the databases used in the experiments

Name	Number of Examples	Number of Attributes	Number of Classes	Percentage of Boolean attributes	Abbreviation
Lenses	24	5	3	60	Len
Soybean (small)	47	36	4	36.11	Soy
Postoperative Patient Data	87	9	3	11.11	Pos
Promoters	106	58	2	1.7	Pro
Hayes-Roth	132	5	3	0	Hay
Monks1	432	7	2	43	Mk1
Monks2	432	7	2	43	Mk2
Monks3	432	7	2	43	Mk3
Balance	625	5	3	0	Bal
Tic-tac-toe	958	10	2	11.11	Tic
CMC	1473	10	3	30	Cmc
Car	1728	7	4	0	Car
Splice	3190	61	3	0	Spl
Kr-vs-kp	3196	36	2	97.3	Krs
LED	6000	8	10	87.5	Led
Nursery	12960	9	5	11.11	Nrs
Kr-V-K	28056	7	17	0	Krv
Synt256	256000	21	10	0	Syn

The databases originally are complete, and missing data were introduced randomly. This enables computing performance index, in terms of accuracy of imputation, defined as the number of correct imputations over the total number of missing values, by comparing imputed values with the original values. Missing values were introduced uniformly into all attributes, except the class attribute. The missing values were introduced at six different levels, i.e. 5%, 10%, 20%, 30%, 40% and 50% to demonstrate the impact of the amount of missing data on the imputation quality.

The experimental section is divided into three parts:

1. *Framework module evaluation.* The goal is to provide motivation for the proposed design of the framework. The effect of each of the three framework modules on the accuracy of imputation improvement is experimentally demonstrated. Figure 22 shows how the experimental evaluation was performed for each of the modules, and for the entire framework.

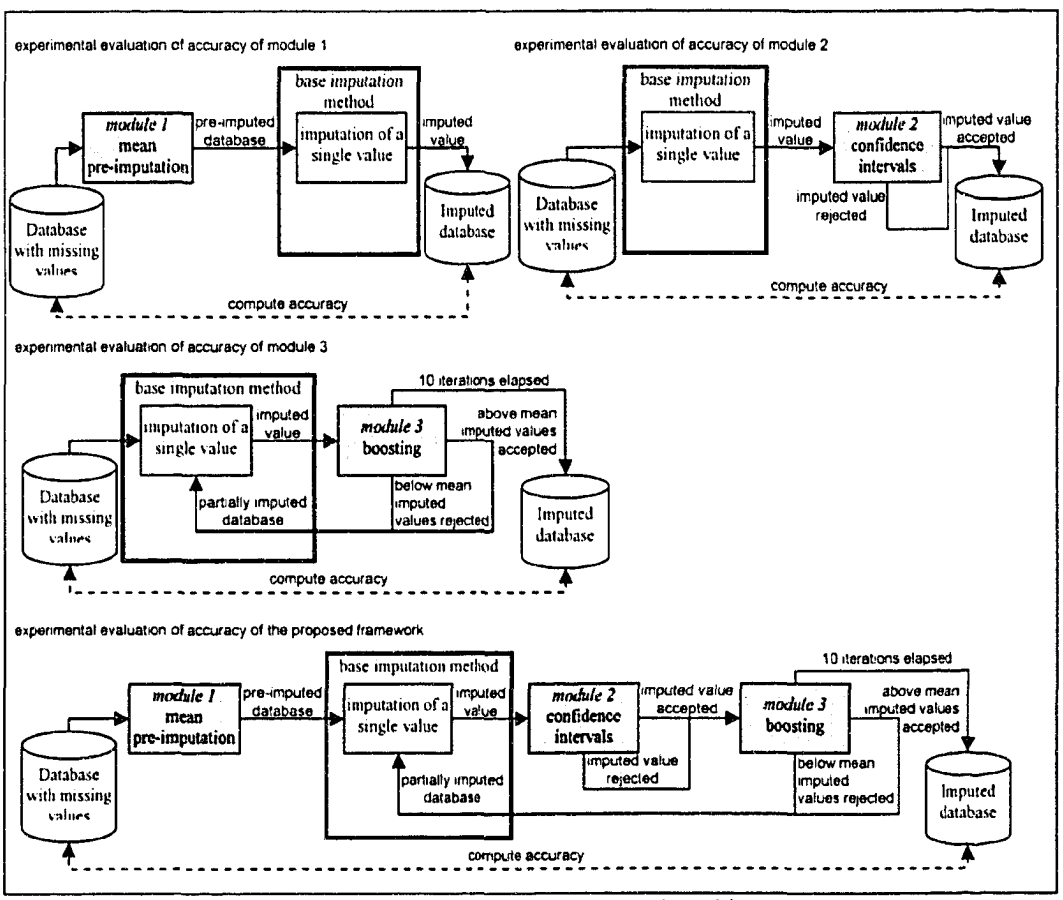


Figure 22. Experimental evaluation of the proposed framework and its components

2. *Experimental comparison with other imputation methods.* The goal is to experimentally compare quality of imputation between the standalone base methods,

i.e. Naïve-Bayes ML based imputation and hot deck, base methods in the framework, and other state of the art imputation methods.

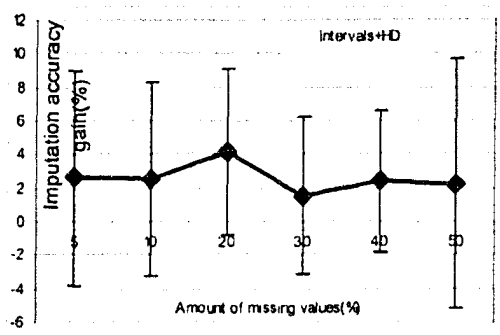
3. *Experimental complexity analysis*. The goal is to show that computational complexity of application of the proposed framework is linear, and therefore does not worsen complexity of the base method. Running times for both base methods, and base methods in the framework are compared between each other and with the theoretical complexity estimates.

5.2.2 Framework Modules Evaluation

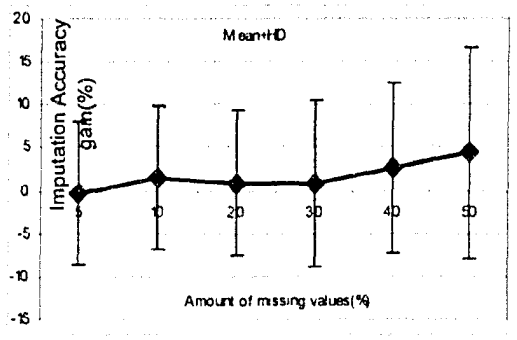
This chapter summarizes experiments that apply the mean pre-imputation, confidence intervals and boosting modules of the framework in separation to show accuracy gain that corresponds to each of the modules. The experiments compare accuracy of imputation of the two considered base methods (hot-deck (HD) and Naïve-Bayes ML based (NB)), the base methods with each of the framework's modules in separation, and finally the base methods combined with the entire framework. They are performed on fifteen databases (*syn* database is omitted) with six different levels of missing values. The results report average (over all datasets) imputation accuracy gain, defined as the difference between the imputation accuracy of base method with one of the framework's modules or the entire framework and the imputation accuracy of the base method, for all considered levels of missing data.

Figure 23 shows the results for the hot deck imputation as the base imputation method. Figure 23a shows that applying confidence intervals results in average imputation accuracy gain of up to 4%. Figure 23b shows that using the mean pre-imputation results in imputation accuracy gain by up to 4.5%, and that the improvements are larger for

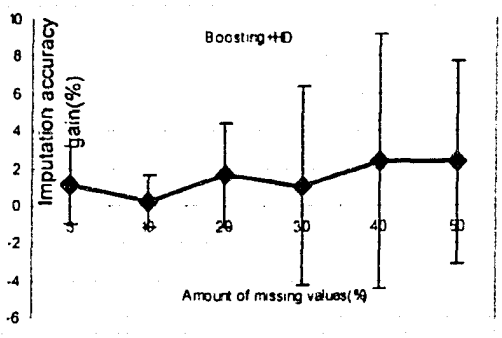
larger amounts of missing data. This is related to poorer imputation accuracy of the base method with increasing amount of missing data, which is compensated by better effectiveness of the framework's module. Figure 23c shows impact of boosting, which improves imputation by up to 2.5%, and is also characterized by increasing trend. Finally, the average imputation accuracy gain for the entire framework is shown in Figure 23d. It is evident that using the proposed framework results, on average, in raising the accuracy of imputation by up to 9%, which is a significant improvement.



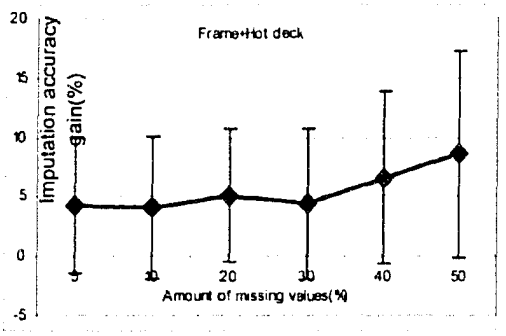
a) Improvement by using Confidence Intervals module



b) Improvement by using Mean Pre-Imputation module



c) Improvement by using Boosting module



d) Improvement by using entire framework

Figure 23. Improvement in imputation accuracy for hot deck base imputation method (vertical bars show standard deviation values)

We note that the individual effects of all modules are not cumulative, but the overall improvement shown by the framework is significantly larger than the improvements

generated by each of the modules, and close to be cumulative. In addition, the increasing trend in improvement with the increasing amount of missing values shows that the framework can effectively compensate for the degradation of accuracy of the base method.

Figure 24 shows detailed results concerning the difference between imputation accuracy of the framework with hot deck as the base method, and standalone hot deck imputation for six levels of missing values and fifteen databases. The bars with black ceiling represent negative values, which result from decrease of imputation accuracy related to application of the framework, while gray ceilings show improvement. It is clear that for most datasets and different levels of missing data the imputation accuracy was improved by applying the framework: 13 times accuracy was worse, while 47 times it was improved. We note that the highest improvements were about 25%.

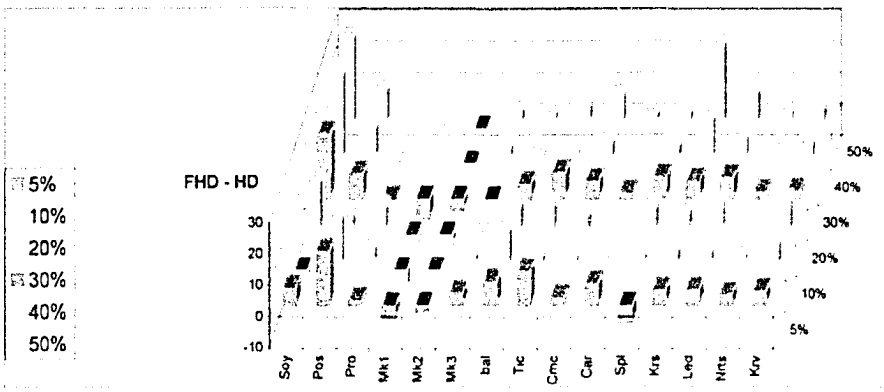
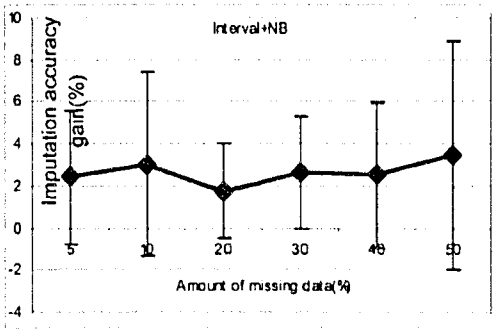


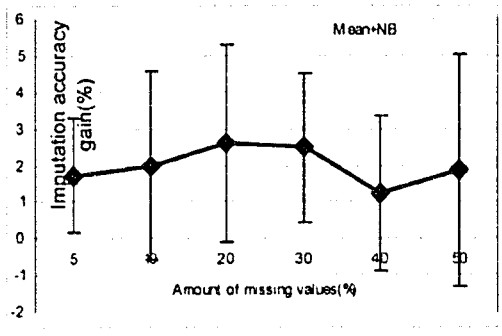
Figure 24. Difference in imputation accuracy between the hot deck with and without the framework (FHD - HD) for six levels of missing values and for fifteen databases

In the following graphs, a similar analysis is applied when using Naive-Bayes ML imputation method as the base method. The average improvement in accuracy of imputation and its standard deviation are summarized in Figure 25. Figure 25a shows the impact of confidence interval, which results in imputation accuracy gain by up to 3.5%.

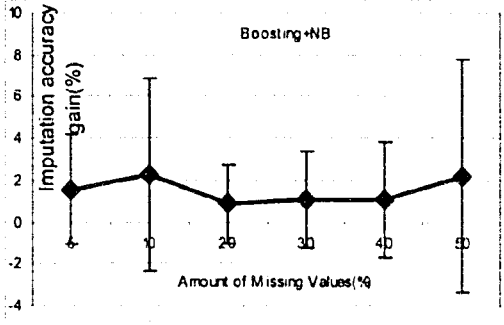
Similarly, Figure 25b shows average improvement of imputation accuracy due to applying mean pre-imputation, and Figure 25c due to boosting modules. In both cases the achieved imputation accuracy gain ranges between 1% and 2.5%. Finally, Figure 25d shows that application of all modules in tandem results in imputation accuracy gain up to 4%. Although the effects of all modules are not cumulative, the overall improvement is significantly larger than the improvement resulting from application of the best module.



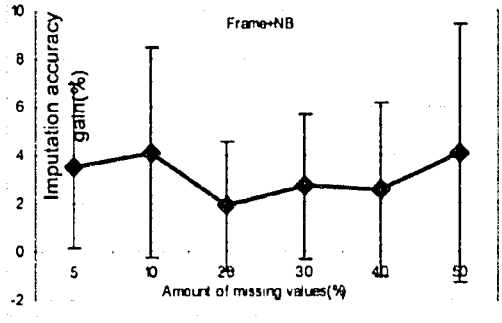
a) Improvement by using Confidence Intervals module



b) Improvement by using Mean Pre-Imputation module



c) Improvement by using Boosting module



d) Improvement by using entire framework

Figure 25. Improvement in imputation accuracy for Naïve Bayes ML base imputation method (vertical bars show standard deviation values)

Figure 26 shows the difference in imputation accuracy between the Naïve-Bayes ML imputation method with and without the framework for six levels of missing values and for fifteen datasets. Similar to results shown in Figure 24, they show that majority of the

values (53 out of 60) are positive, which indicates improvement in accuracy of imputation as a result of using the framework.

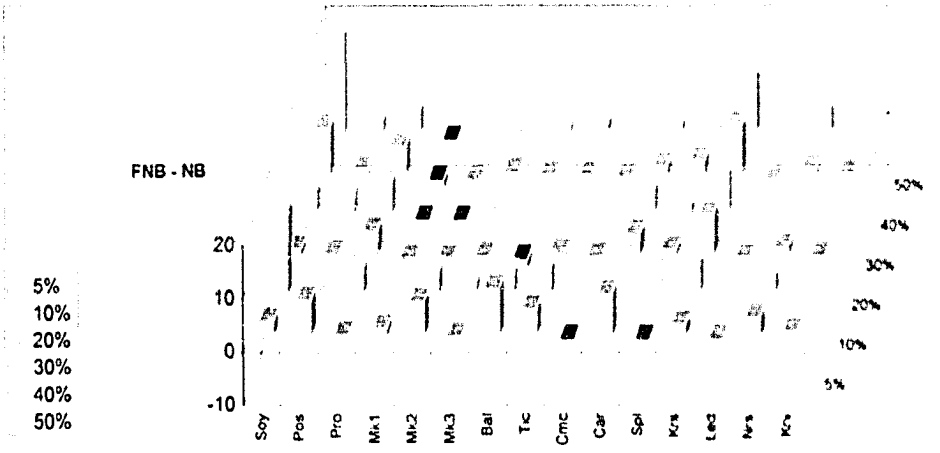


Figure 26. Difference in the accuracy of imputation between the Naïve-Bayes ML imputation method with and without the framework (FNB - NB) for six levels of missing values and for fifteen databases

We conclude that application of each of the framework's modules in separation and together always results in average improvement of imputation accuracy for both of the considered base imputation methods. It can be expected that application of the framework should on average result in improving the imputation accuracy. In the next section, the amount of the improvement is quantified and compared with performance of other imputation methods. In general, we note that the amount of imputation accuracy gain depends on the performance of the base method, i.e. it is larger for low quality imputation methods such as hot deck, while being smaller for better quality base methods such as Naïve Bayes ML method.

Figure 27 and Figure 28 compare average accuracy of imputation of the hot deck and Naïve Bayes ML imputation methods with and without the framework, respectively.

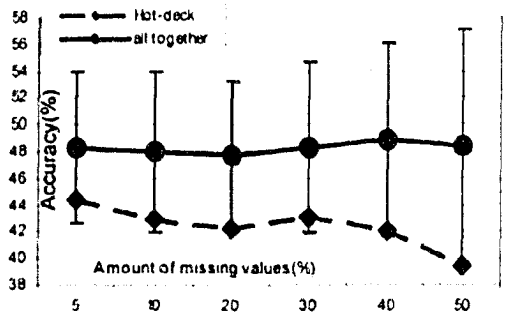


Figure 27. Accuracy of imputation using framework with hot deck and standalone hot deck

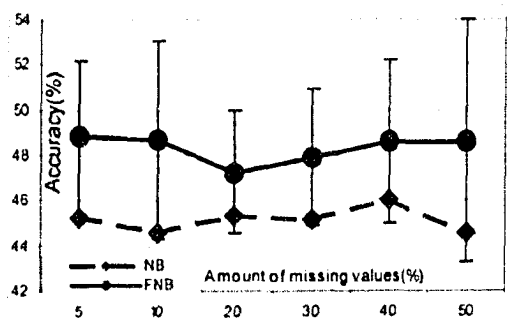


Figure 28. Accuracy of imputation using framework with Naive-Bayes ML and standalone Naive-Bayes ML method

The comparison shows that application of the framework results in flattening the accuracy curve with respect to increasing the amount of missing data, especially for hot deck imputation method, see Figure 27. Application of the proposed framework compensates for degradation of imputation accuracy of the base method caused by larger amount of missing information, which is valuable when dealing with sparse databases. We again note that the amount of obtained improvement depends on the quality of the base method. For high quality method, such as Naïve-Bayes ML imputation, the improvement is relatively small, i.e. 2% - 4%, while the accuracy of the base method is on average about 44.5%. In case of hot deck imputation, the improvement ranges between 4 and 9%, while the accuracy of the base method is on average about 42%. We note that the accuracy of both imputation methods combined with the framework is very similar.

We stress that a recent study as well as this thesis have shown Naïve Bayes ML method has superior accuracy compared with hot deck imputation [28]. At the same time, application of framework to hot deck method results in imputation method that has higher accuracy than accuracy of standalone Naïve Bayes ML based imputation method. This shows that the proposed framework provides a solution that helps to develop relatively simple and efficient imputation methods that are characterized by high imputation accuracy.

5.2.3 Experimental Comparison with Other Imputation Methods

Several representative imputation methods from the three categories are chosen for the experimental part. They include data driven methods, such as random sampling multiple imputation (SAM), mean (Mean) and hot deck (HD), model based methods, such as regression including polytomous and logit multiple imputation (POLYLOGREG), and linear discriminant analysis and logit multiple imputation (LDALOGREG), and ML based methods, such as Naïve Bayes (NB). These methods are compared with Naïve Bayes based and hot deck imputations combined with the proposed framework (FNB and FHD, respectively). Therefore total of 8 methods are compared on 15 databases. The multiple imputation methods were set to 5 imputation rounds. The number of round was established experimentally. More rounds resulted, on average, in insignificant or no improvement in accuracy, but worsened the running time.

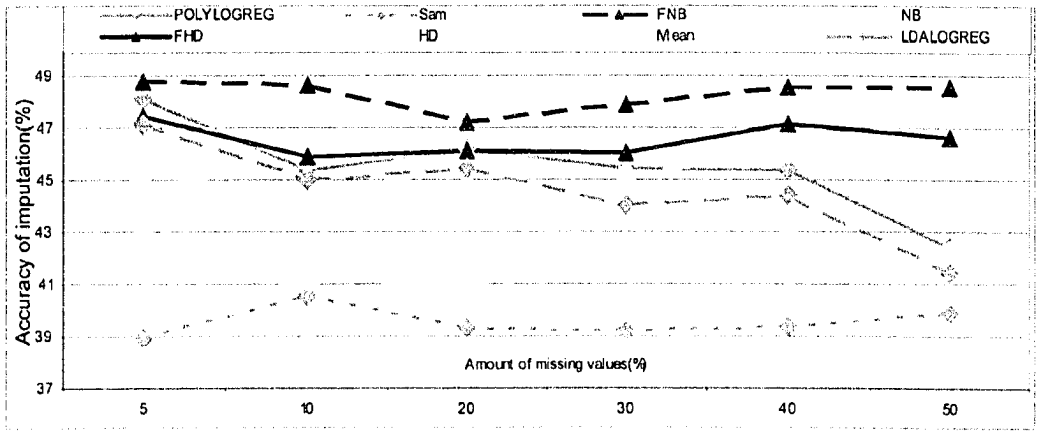


Figure 29. Summary of imputation accuracy results for the considered eight imputation methods

Figure 29 shows average, over fourteen datasets (*syn* database is omitted and *soy* database has not enough records to perform regression based imputation), imputation accuracy using the eight imputations and for all considered levels of missing values. The results show that the best results are achieved by the FNB method. The method is consistently better considering the entire spectrum of missing values levels. The second best is the FHD imputation, which has superior accuracy over more complex model base imputation methods, such as POLYLOGREG and LDALOGRED, and the ML based NB imputation, for larger amount of missing values, and similar accuracy for small amounts. The least accurate are the data driven imputation methods, such as HD, Mean, and SAM. We note that while the HD imputation has poor performance, applying the framework results in improving accuracy to be superior to, or at least as accurate as accuracy of advanced model based methods. We also note that accuracy of some imputation methods, such as LDALOGREG, POLYLOGREG, and SAM deteriorates with the increasing amount of missing data, while the methods that utilize the framework perform with the same level of accuracy. The experiments clearly demonstrate effectiveness of the

proposed framework, which can be applied to simple imputation methods to provide significant improvement in imputation accuracy, and can help to achieve accurate imputations even in the presence of large quantities of missing information.

In the scatter plot of Figure 30 the accuracy of the FNB imputation method is compared with accuracy of all methods that do not utilize the proposed framework. The shown values are the average, over the six levels of missing values, imputation accuracy for each of the fourteen databases. The y-axis position is the accuracy of FNB, while the x-axis is the accuracy of other imputation method. Therefore, points above diagonal line correspond to databases for which FNB achieves better average imputation accuracy. Visual inspection confirms that the FNB imputation method performs better than other imputation methods on significant majority of the databases. Similarly, a scatter plot of Figure 31 compares FHD method with other methods that do not utilize the proposed framework. Again, since majority of the points are located above the diagonal line, we conclude that FHD method on average performs better than other imputation methods.

In the nutshell, experimental results indicate that application of the proposed framework results in improvement of imputation accuracy when compared with accuracy of the standalone base imputation method and other state of the art single and multiple imputation methods. Applying the framework to simple imputation methods, such as hot deck, results in an imputation method that on average performs better than complex model based imputation methods. We also note that application of the framework makes the base method more robust to the larger levels of missing data.

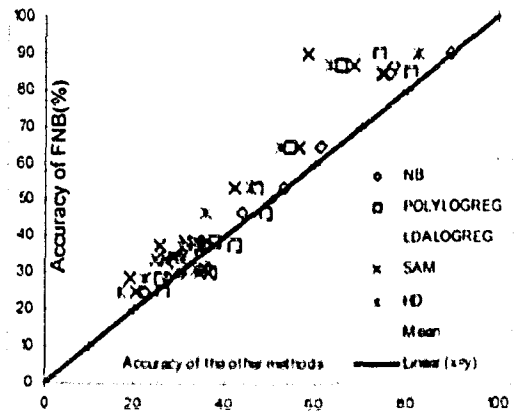


Figure 30. Accuracy of imputation using the framework with Naive-Bayes against other imputation methods

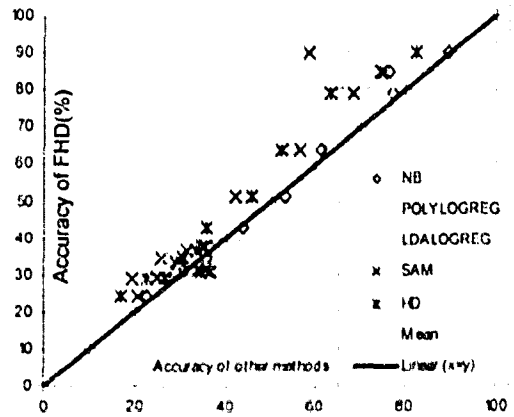


Figure 31. Accuracy of imputation using the framework with Hot deck against other imputation methods

5.2.4 Analysis of Experimental Complexity

The demonstrated experiments show that the application of the proposed framework results in improving imputation accuracy. However the important question is related to how much computational effort is necessary to apply the framework and most importantly if the application of the framework could worsen computational complexity of the base imputation method. Therefore, following tests aim to test computational complexity associated with application of the proposed framework to a base imputation methods. The main goal is to experimentally assess theoretical estimate that implies linear complexity with respect to the number of records. Confirming this hypothesis implies that the application of the framework does not worsen asymptotic complexity of the base imputation method, since there is no imputation method with sub-linear complexity. For this purpose, the *syn* database with 256K records was chosen to observe steepness of the running time curve with increasing size of the database. The *syn* database

was used to randomly derive nine databases of different sizes including 1K, 2K, 4K, 8K, 16K, 32K, 64K, 128K, and finally the original database with 256K records. The experiments record running time on the databases with the incrementally doubled size. Also to investigate the effect of level of missing values on the asymptotic complexity of the method, two levels of missing values, i.e. 10% and 60%, were randomly introduced into the databases, and the experiments were performed separately for both levels. Figure 32 shows the run time versus size of the database in the log-log scale for FNB and NB imputation methods and two levels of missing values (FNB 10%, FNB 60%, NB 10%, and NB 60%) and for the generated nine databases. Both linear and log-linear curves were plotted on the figure for the reader's convenience.

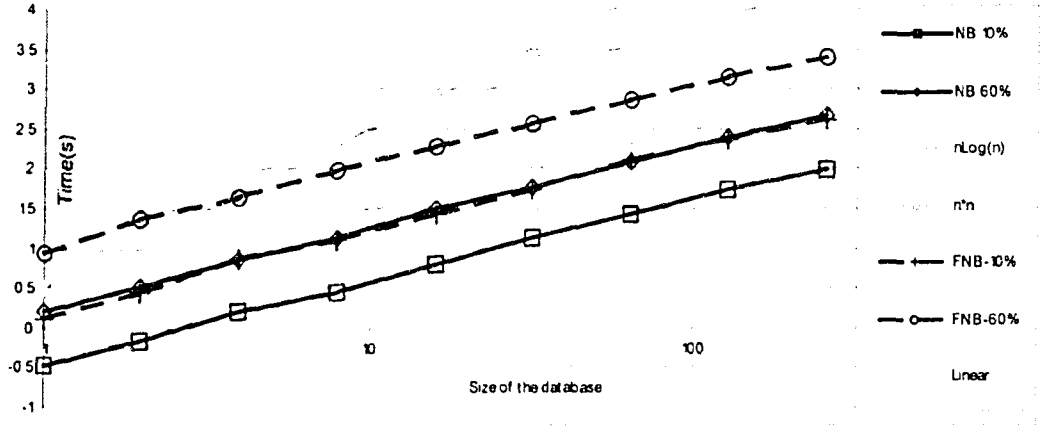


Figure 32. Runtime against the size of the database for the FNB and NB imputation methods and for 10% and 60% of missing values

The curves for FNB and NB methods align in parallel to the linear curves for both levels of missing data, which shows that the linear asymptotic complexity of the Naïve Bayes ML based imputation method is preserved when applying the proposed framework. We

note that the corresponding curves for the standalone and framework based method are shifted in parallel. This indicates that additional computational work, which is connected with application of the framework, is performed, but it does not change the type of asymptotic complexity.

Similar experiment was performed with FHD and HD imputation methods and summarized in Figure 33. Closer analysis of the figure shows that plots for both HD and FHD are parallel to the quadratic curve. This implies that original complexity of the standalone hot deck method, i.e. quadratic with the number of data records, is preserved when the framework is applied. The corresponding curves for the standalone and framework based method are shifted in parallel, which indicates identical asymptotic complexity, but the results show that application of the framework actually shortens the running time when compared to the running time of the standalone method. This is the results of applying confidence intervals that filter out less probable candidates for imputed values. As a result, the search space of the hot deck imputation procedure to find the closest record is reduced resulting in a shorter running time. Therefore, in case of the hot deck imputation, application of the framework not only results in improved imputation accuracy, but also in lowering running time of the method.

In short, application of the framework does not change the asymptotic complexity of the base method, while it results in increasing the accuracy of imputation.

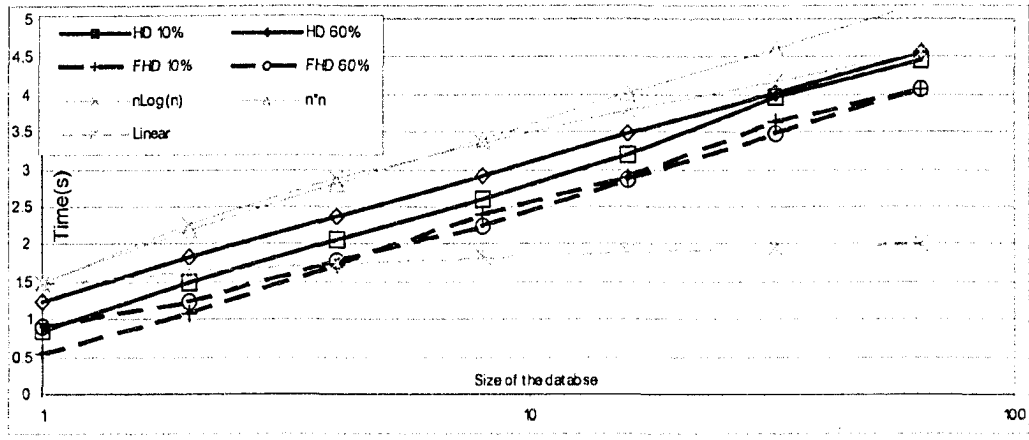


Figure 33. Run time against the size of the database for Hot deck imputation with and without the framework, for 10% and 60% of missing values

The experimental complexity analysis is supplemented by the running time of the eight considered missing data imputation methods, for the fourteen databases and for the six levels of missing values, see Table 3. First two rows for each of the missing data levels show results of imputation methods that use the proposed framework, i.e. FNB and FHD, next three rows show results for multiple imputation methods, i.e. LDALOGREG, LDALOGREG, and SAM, and last three rows show results for single imputation methods, i.e. NB, HD, and Mean. Bolded values indicate the lowest run time for a given database with a given level of missing data. As expected, the mean imputation is the fastest imputation method, while at the same time, as Figure 29 shows, its imputation accuracy on average is better than accuracy of SAM and HD methods.

We note that while in general high amounts of missing values result in lowering imputation accuracy, the mean imputation method is robust to the large amount of missing values [28].

Table 3. Running time of the eight imputation methods, for the fourteen databases and the six levels of missing values

		Pos	Pro	Mk1	Mk2	Mk3	Bal	Tic	Cmc	Car	Spl	Krs	Led	Nrs	Krv
5%	FNB	0.00	0.11	0.03	0.02	0.02	0.00	0.08	0.28	0.10	11.31	4.13	0.37	2.23	3.80
	FHD	0.02	0.04	0.33	0.11	0.15	0.26	0.74	2.44	2.39	616.50	40.56	45.55	1351.11	721.03
	POLYLOGREG	10.04	1317.95	12.54	13.42	13.27	22.97	99.99	1201.94	126.17	714.56	71.25	62079.59	1312.23	11887.99
	LDALOGREG	5.32	306.96	6.42	6.11	6.11	6.49	99.99	117.29	28.77	707.00	20.47	3310.00	301.47	1080.00
	SAM	3.50	74.19	3.00	2.98	3.19	2.89	54.70	52.01	13.58	444.55	8.70	1978.00	177.73	371.00
	NB	0.01	0.03	0.00	0.00	0.00	0.00	0.01	0.07	0.03	1.59	0.59	0.11	0.40	0.90
	HD	0.01	0.15	0.25	0.15	0.14	0.29	1.83	2.17	3.96	364.60	130.75	58.12	907.67	490
	Mean	0.00	0.02	0.00	0.02	0.00	0.00	0.00	0.02	0.00	0.28	0.14	0.06	0.13	0.28
10%	FNB	0.01	0.20	0.04	0.04	0.03	0.03	0.11	0.36	0.19	14.95	5.00	0.48	2.71	4.54
	FHD	0.03	0.09	0.25	0.28	0.32	0.44	1.90	14.11	13.82	600.00	34.16	94.83	1525.99	961.00
	POLYLOGREG	8.70	1374.00	12.63	11.92	13.08	21.81	98.10	1149.67	115.94	699.72	78.55	50357.00	1294.41	12756.00
	LDALOGREG	5.73	297.73	6.67	6.47	6.33	6.64	98.10	111.45	30.06	649.39	21.14	3218.00	273.40	1087.56
	SAM	3.30	76.90	2.99	3.15	2.94	2.80	56.20	44.88	13.95	153.03	8.96	1821.00	178.00	359.00
	NB	0.00	0.04	0.02	0.00	0.00	0.01	0.03	0.11	0.04	2.03	0.75	0.16	0.52	1.50
	HD	0.03	0.13	0.28	0.29	0.26	0.53	4.08	9.50	10.19	427.33	139.00	104.17	1051.95	738
	Mean	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.03	0.02	0.31	0.14	0.06	0.14	0.30
20%	FNB	0.01	0.36	0.05	0.06	0.05	0.04	0.17	0.67	0.25	19.59	5.88	0.69	3.47	6.39
	FHD	0.06	0.19	0.44	0.46	0.30	1.12	4.14	12.61	6.73	439.37	86.71	104.34	975.95	1325.84
	POLYLOGREG	7.97	1393.50	13.49	11.92	12.48	20.88	97.47	1064.78	112.00	620.36	59.11	46806.00	1076.13	11253.00
	LDALOGREG	3.92	270.83	7.03	6.47	6.27	6.59	97.47	101.72	28.97	599.12	21.51	1607.00	250.57	1461.99
	SAM	3.36	74.60	3.17	3.14	3.22	2.92	63.31	45.97	14.03	336.86	8.49	1955.00	192.55	416.95
	NB	0.00	0.06	0.02	0.02	0.02	0.02	0.06	0.19	0.08	2.89	0.99	0.25	0.89	3.81
	HD	0.03	0.09	0.35	0.34	0.31	0.75	6.72	8.08	6.78	378.25	140.74	103.64	1094.03	1136.00
	Mean	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.04	0.00	0.29	0.16	0.06	0.13	0.31
30%	FNB	0.02	0.50	0.06	0.08	0.06	0.08	0.24	1.03	0.35	24.58	6.78	0.83	4.45	7.39
	FHD	0.07	0.18	0.48	0.58	0.36	0.98	3.51	11.58	6.68	135.42	86.22	102.48	1165	5942.00
	POLYLOGREG	7.18	1467.91	11.27	11.40	11.66	18.36	95.01	932.94	102.24	563.49	51.50	37999.71	969.40	9648.00
	LDALOGREG	3.85	248.51	7.07	6.40	6.49	7.07	95.01	86.05	29.34	526.98	21.68	1844.00	258.19	973.45
	SAM	3.35	76.83	2.97	3.42	2.98	2.84	58.4	48.90	19.81	340.07	8.69	1844.00	185.33	384.00
	NB	0.00	0.07	0.01	0.03	0.02	0.03	0.08	0.27	0.11	3.72	1.19	0.31	1.21	4.06
	HD	0.02	0.11	0.32	0.33	0.28	0.82	2.84	5.20	5.53	170.95	131.11	78.40	1165.00	1585.00
	Mean	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.03	0.00	0.29	0.18	0.07	0.14	0.31
40%	FNB	0.02	0.62	0.07	0.08	0.06	0.11	0.33	1.26	0.41	32.41	8.86	1.01	5.93	9.44
	FHD	0.05	0.18	0.47	0.59	0.47	0.97	3.08	18.64	6.21	303.77	98.52	96.01	1057.00	5521.00
	POLYLOGREG	6.34	1456.96	11.06	10.35	11.55	16.39	82.48	862.94	88.72	426.67	47.97	32124.69	852.56	8022.00
	LDALOGREG	3.89	232.92	7.02	6.61	6.64	7.47	82.48	58.81	30.46	474.25	24.08	1593.00	241.77	907.92
	SAM	3.17	76.69	3.11	3.25	3.09	2.98	54.75	59.44	14.19	324.00	8.61	1593.00	172.81	321.49
	NB	0.00	0.09	0.04	0.03	0.03	0.03	0.09	0.30	0.14	4.58	1.61	0.35	1.54	5.16
	HD	0.02	0.12	0.26	0.25	0.26	0.55	1.7	3.26	4.99	176.22	287.75	56.05	1057.00	1203.00
	Mean	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.02	0.02	0.31	0.16	0.07	0.16	0.32
50%	FNB	0.02	0.75	0.11	0.10	0.09	0.17	0.40	1.54	0.52	37.05	9.42	1.14	3.75	12.11
	FHD	0.04	0.19	0.62	0.47	0.46	0.95	2.91	17.31	6.41	195.25	81.88	97.13	566.00	5923.00
	POLYLOGREG	6.61	1210.97	10.29	10.64	10.79	15.63	85.50	749	96.15	383.88	45.47	28000.00	1085.77	6069.00
	LDALOGREG	3.10	236.85	7.18	6.76	6.54	7.61	85.50	51.68	30.05	380.00	24.36	1689.00	263.15	858.40
	SAM	3.19	69.02	3.08	3.39	3.06	2.88	60.59	52.70	14.58	343.75	8.83	1689.00	176.84	318.65
	NB	0.00	0.11	0.03	0.03	0.03	0.04	0.13	0.42	0.16	5.17	1.58	0.42	0.83	7.34
	HD	0.02	0.13	0.24	0.25	0.17	0.39	1.87	3.61	3.46	170.52	354.00	41.84	566.00	791.00
	Mean	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.03	0.02	0.32	0.18	0.06	0.14	0.34

Close analysis of the above table reveals that:

- Running time of the most accurate FNB method, which uses the proposed framework, is always significantly shorter than the running time of the considered multiple imputation methods, with exception of results for the *krs* database for large amounts

of missing data that are a bit better than the results of poorly performing SAM method.

- Application of the proposed framework to NB method results on average, through all experiments, in 3.7 times increase of running time when compared with running time of the standalone method. Similarly for the *HD* method, 1.6 times increase is suffered.
- Application of the framework to the hot deck imputation method may result in decrease of the running time when compared with the running time of the standalone method. It can be observed for the *krs* database, and for small amount of missing data for the *pro*, *bal*, *tic*, *car*, and *led* databases. This is attributed to the filtering of less probable candidate imputed value by the confidence intervals, which results in shorted time to find the closest record. Consequently computational time may be reduced.

The regression based multiple imputation method, POLYLOGREG, is characterized by the longest running time. Its running time is 6 orders of magnitude slower than running time of the fastest single Mean imputation method, and 5 orders of magnitude slower than running time of the most accurate FNB single imputation method; see results for the *led* database. Although the experiments were performed using the same hardware and different software packages (the POLYLOGREG, LDALOGREG, and SAM methods were executed using MICE package [40], while the remaining methods were implemented in C++ by the authors), which may result in some minor distortion of running time results, the significance of the difference cannot be disputed.

To summarize, the experiments demonstrate that each module of the proposed framework i.e., confidence intervals, mean pre-imputation and boosting, improves imputation accuracy of the base imputation method. The proposed framework can be successfully used to improve imputation accuracy of any base method, which can generate weights representing quality of each imputed value to perform boosting. In practice almost all existing imputation methods satisfy this requirement; this paper demonstrates how to apply the framework with two imputation methods: hot deck and Naïve Bayes ML based imputation. Application of the framework results, on average, with significant gain of imputation accuracy when compared with accuracy of the base method. The results show that a poor quality single imputation method, such as hot deck, can be improved with the use of the framework to match quality of advanced multiple imputation methods. The results also show that the best imputation accuracy was achieved by the Naïve Bayes ML based imputation in the proposed framework. It performed with higher imputation accuracy and in the lower running time than any of the considered model based and ML based single and multiple imputation methods. Finally, we have shown, both theoretically and experimentally, that the proposed framework has linear asymptotic complexity, and therefore its application does not worsen asymptotic computational complexity of the base method.

6. Summary and Conclusions

Most of the real world industrial and research databases have a shortcoming of containing missing values. One of common ways to cope with this problem is to perform imputation (filling in) of the missing values through variety of statistical and machine learning (ML) procedures. In this study, an experimental comparison of several imputation methods is performed to evaluate the performance of each method. Also, a novel framework that aims to improve accuracy of the existing imputation methods is proposed. The new framework consists of three components: mean pre-imputation, confidence intervals and boosting, and can be applied to many of existing imputation methods, including data driven, model based, and ML based. The framework is characterized by a number of advantages. Its application to an imputation methods results, on average, in significant improvement of imputation accuracy, while at the same maintaining the same asymptotic computational complexity. For some imputation methods, such as hot deck, application of the framework may even result in lowering the running time, while in general computational cost of applying the framework is relatively low.

To demonstrate advantages of the proposed framework, it was used with two imputations methods: Naïve-Bayes ML based imputation method and hot deck data driven imputation method. The two above imputation methods were experimentally tested on fifteen databases, and compared with six other popular imputation methods, including single imputation mean and hot deck methods, multiple imputation random sample, regression, and linear discriminant analysis methods, and ML based single imputation Naïve Bayes method. The results show that a significant improvement of imputation accuracy can be

achieved by applying the proposed framework, and that the accuracy of the framework based methods was on average the highest among the considered methods. We stress that application of the proposed framework to a simple and low quality single imputation method, such as hot deck, resulted in a method that was characterized by imputation accuracy comparable to accuracy of advanced multiple imputation methods. At the same time, application of the framework to a quality imputation method, such as the ML based Naïve Bayes method, resulted in imputation accuracy that was superior with respect to accuracy of other imputation methods. We also show, both theoretically and experimentally, that application of the proposed framework has linear complexity, and therefore does not change asymptotic complexity of the associated imputation method.

References

[1] Allwein, E.L., Schapire, R.E., and Singer, Y., Reducing Multiclass to Binary: A Unifying Approach for Margin Classifiers, *Journal of Machine Learning Research*, vol. 1, pp. 113-141, 2000

[2] Alzola, C., and Harrell, F., *An Introduction Of S-Plus and The Hmisc and Design Libraries*, (download from http://www.med.virginia.edu/medicine/clinical/hes), 1999

[3] Barnard, J., and Meng, X.L., Applications of Multiple Imputation in Medical Studies: from AIDS to NHANES, *Statistical Methods in Medical Research*, vol.8, pp.17-36, 1999

[4] Blake, C.L., and Merz, C.J., *UCI Repository of Machine Learning Databases*, [http://www.ics.uci.edu/ ~mlearn/MLRepository.html], Irvine, CA: U. of California, Department of Information and Computer Science, 1998

[5] Brand, J.P.L., *Development, Implementation and Evaluation of Multiple Imputation Strategies for The Statistical Analysis of Incomplete Datasets*, Ph.D. thesis, Erasmus University Rotterdam, 1999

[6] Brazdil, P., Gama, J., and Henery R., Characterizing the Applicability of Classification Algorithms Using Meta Level Learning, In: F. Bergadano, and L. de Raedt (Eds.), *Proceedings of the European Conference on Machine Learning* (ECML-94), pp.83-102, 1994

[7] Breiman, L., Prediction Games and Arcing Algorithms, *Neural Computation*, vol. 11(7), pp. 1493-1518, 1999, Also Technical report 504, Statistics Department, University of California Berkeley.

[8] Buck, S.F., A Method of Estimation of Missing Values in Multivariate Data Suitable for Use with an Electronic Computer, *Journal of Royal Statistical Society*, vol. B22, pp.302-306, 1960

[9] Buuren, S., and Oudshoorn, CGM., *Flexible Multivariate Imputation by MICE*, Leiden: TNO Preventie en Gezondheid, TNO/VGZ/PG 99.054, 1999

[10] Buuren, S., and Oudshoorn, CGM., *Flexible Multivariate Imputation by MICE*, Leiden: TNO Preventie en Gezondheid, TNO/VGZ/PG 99.045, 1999

[11] Buuren, S.V, Mulligen, E.M., and Brand, J.P.L., Routine Multiple Imputation In Statistical Databases, In: French JC, Hinterberger H, eds. *Proceedings of the Seventh International Working Conference on Scientific and Statistical Database Management, IEEE Computer Society Press*, Los Alamitos, CA, pp.74-78, 1994

[12] Campbell, C., and Bennett, K.P., A Linear Programming Approach to Novelty Detection, In Leen, T.K., Dietterich, T.G., and Tresp, V., Editors, *Advances in Neural Information Processing Systems*, vol. 13, pp. 395-401, MIT Press, 2001

[13] Casella, G., and George, E.L., Explaining the Gibbs Sampler, *American Statistician*, vol.46, pp.167-174, 1992

[14] Chan, K., Lee, T.W., and Sejnowski, T.J., Variational Bayesian Learning of ICA with Missing Data, *Neural Computation*, vol.15:8, pp.1991-2011, 2003

[15] Cheeseman, P., Kelly, J., Self, M., Stutz, J., Taylor, W., and Freeman, D., AutoClass: a Bayesian Classification System, *Proceedings of the Fifth International Workshop on Machine Learning*, Ann Arbor, pp. 54-64, 1988

[16] Cios, K. J., and Kurgan, L.A., Hybrid Inductive Machine Learning: An Overview of CLIP Algorithms, In: Jain, L.C., and Kacprzyk, J. (Eds.), *New Learning Paradigms in Soft Computing*, pp. 276-322, Physica-Verlag (Springer), 2001

[17] Cios, K.J., and Kurgan, L.A., CLIP4: Hybrid Inductive Machine Learning Algorithm that Generates Inequality Rules, *Information Sciences*, Special Issue on *Soft Computing Data Mining*, Pal, S.K., and Ghosh, A. (Eds.), 163:1-3, pp. 37-83, 2004

[18] Cios, K.J., and Moore, G., Uniqueness of Medical Data Mining, *Artificial Intelligence in Medicine*, 26:1-2, pp.1-24, 2002

[19] Cios, K.J., Pedrycz, W., and Swiniarski, R., *Data Mining Methods for Knowledge Discovery*, Kluwer Academic Publishers, 1998

[20] Conversano, C., and Capelli, C., Missing Data Incremental Imputation through Tree-based Methods, in Hardle W. e Ronz B. (eds.), *Proceedings in Computational Statistics (COMPSTAT 2002)*, Physica-Verlag, pp.455- 460, 2002

[21] Dempster, A.P., Laird, N.M., and Rubin, D.B., Maximum Likelihood from Incomplete Data via the EM Algorithm (with discussion), *Journal of Royal Statistical Society*, vol.82, pp.528-550, 1978

[22] Dettling, M., and Buhlmann, P., Boosting for tumor classification with gene expression data, In *Bioinformatics*, vol. 19(9), pp. 1061-1069, 2003

[23] Domingo, C., and Watanabe, O., A modification of AdaBoost. *Proceedings in Computational Learning Theory (COLT 2000)*, San Francisco, Morgan Kaufman, pp. 180-189, 2000

[24] Drucker, H., Schapire, R.E., and Simard, P.Y., Boosting Performance in Neural Networks. *International Journal of Pattern Recognition and Artificial Intelligence*, vol.7, pp. 705-719, 1993

[25] Duda, R.O., and Hart, P.E., Pattern *Classification and Scene Analysis*, John Wiley, 1977

[26] Duffy, N., and Helmbold, D.P., Potential Boosters?, In Solla, S.A., Leen, T.K., and Muller, K.R., editors, *Advances in Neural Information Processing Systems*, vol. 12, pp. 258-264, MIT Press, 2000

[27] Farhangfar, A., Kurgan, L., & Pedrycz, W., Novel Framework for Imputation of Missing Values in Databases, *IEEE Transactions on Systems, Man and Cybernetics, Part A*, submitted, 2004

[28] Farhangfar, A., Kurgan, L., and Pedrycz, W., Experimental Analysis of Methods for Handling Missing Values in Databases, *Intelligent Computing: Theory and Applications II Conference*, held in conjunction with *the SPIE Defense and Security Symposium (formerly AeroSense)*, pp.172-182, Orlando, FL, 2004

[29] Feelders, A.J., Handling Missing Data in Trees: Surrogate Splits or Statistical Imputation, *Proceedings of the 3rd European Conference on Principles and Practice of Knowledge Discovery in Data Bases (PKDD99)*, Springer, pp. 329-334, 1999

[30] Ford, B.M., *An Overview of Hot-Deck Procedures. Incomplete Data in Sample Surveys*, Vol. 2. New York: Academic Press, 1983

[31] Frean, M., and Downs, T., *A Simple Cost Function for Boosting*, Technical report, Dep. Of Computer Science and Electrical Engineering, University of Queensland, 1998

[32] Freund, Y., and Schapire, R.E., A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting, *Journal of Computer and System Sciences*, vol. 55, pp. 119-139, Orlando, Academic Press,1997

[33] Freund, Y., and Schapire, R.E., Experiments with a New Boosting Algorithm, *Proceedings of the 13th International Conference on Machine Learning*, pp.148-146, 1996

[34] Freund, Y., and Schapire, R.E., Experiments with a New Boosting Algorithm, *Proceedings in 13th International Conference on Machine Learning*, pp. 146-148, Morgan Kaufmann, 1996

[35] Friedman, J., Hastie, T., and Tibshirani, R.J., Additive Logistic Regression: a Statistical View of Boosting, *Annals of Statistics*, vol. 2, pp. 337-374, 2000, with Discussion pp.375-407, also Technical Report at Department of Statistics, Sequoia Hall, Stanford University

[36] Ghahramani, Z., and Jordan, M.I., Mixture Models for Learning from Incomplete Data, in Greiner, R., Petsche, T., and Hanson, S.J. (Eds.), *Computational Learning Theory and Natural Learning Systems, Volume IV: Making Learning Systems Practical*, MIT Press, pp.67-85, 1997

[37] Grove, A.J., and Schuurmans, D., Boosting in the Limit: Maximizing the Margin of Learned Ensembles, In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pp. 692-699,1998

[38] Hettich, S., and Bay, S. D. The UCI KDD Archive [http://kdd.ics.uci.edu]. Irvine, CA: University of California, Department of Information and Computer Science, 1999

[39] Hopke, P.K., Liu, C., Rubin, D.B., Multiple imputation for multivariate data with missing and below-threshold measurements: time-series concentrations of pollutants in the Arctic, *Biometrics*, vol. 57(1), pp. 22-33, 2001

[40] Horton, N.J., and Lipsitz, S.R., Multiple Imputation in Practice: Comparison of Software Packages for Regression Models with Missing Variables. *The American Statistician*, vol.55 (3): pp.244-254, 2001

[41] Kearns, M., and Mansour, Y., On the Boosting Ability Og Top-Down Decision Tree Learning Algorithms. *Proceedings in 28th ACM Symposium on the Theory of Computing*, pp. 459-468. ACM Press, 1996

[42] Kodratoff, Y., *Introduction to Machine Learning*, Morgan-Kaufmann, 1988

[43] Kurgan, L.A., Cios, K.J., Sontag, M., and Accurso, F.J., Mining the Cystic Fibrosis Data, In: Zurada, J., and Kantardzic, M. (Eds.), *Novel Applications in Data Mining*, IEEE Press, in print, 2004

[44] Lakshminarayan, K., Harp, S.A., and Samad, T., Imputation of Missing Data in Industrial Databases, *Applied Intelligence*, vol 11, pp. 259 – 275, 1999

[45] Langley, P., *Elements of Machine Learning*, Morgan-Kaufmann, 1996

[46] Li, K-H., Imputation using Markov chains, *Journal of Statisicalt Comput Simul.*; vol.30, pp.57-79, 1988

[47] Little, R.J., and Rubin, D.B., *Statistical Analysis with Missing Data*, John Wiley and Sons, 1987

[48] Manly, B.F.J., *Multivariate Statistical Methods: A primer*. Second Edition, Chapman and Hall, London, 1994

[49] Meir, R., El-Yaniv, R., and Ben-David, Shai, Localized Boosting, *Proceedings in Computational Learning Theory (COLT 2000)*, pp. 190-199, San Francisco, Morgan Kaufman, 2000

[50] Michie, D., Spiegelhalterand, D.J., and Taylor, C., *Machine Learning, Neural and Statistical Classification*, Ellis Horwood, 1994

[51] Mitchell, T.M., *Machine Learning*, McGraw-Hill, 1997

[52] Oh, H.L., and Scheuren, F.L., *Weighting Adjustments for Unit Nonresponse, in Incomplete Data in Sample Survey*, Volume 2, Theory and Bibliographies, edited by W.G. Madow, I. Olkin, and D.B. Rubin, Academic Press: New York, pp. 143-183, 1983

[53] Quinlan, J.R., *C4.5: Programs for Machine Learning*, Morgan Kaufman, 1992

[54] Quinlan, J.R., Induction of Decision Trees, *Machine Learning*, vol.1, pp.81-106, 1986

[55] Ratsch, G., Demiriz, A., and Bennett, K., Sparse Regression Ensembles in Infinite and Finie Hypothesis Spaces, *Machine Learning*, vol. 48(1-3), pp. 193-221, 2002, Special Issue on New Methods for Model Selection and Model Combination, Also NeuroCOLT2 Technical Report NC-TR-2000-085

[56] Ratsch, G., Mika, S., Scholkopf, B., and Muller, K.R., Constructing Boosting Algorithms from SVMs: An Application to One-Class Classification, *IEEE PAMI*, vol. 24(9), September 2002

[57] Ratsch, G., Onoda, T., and Muller, K.R., Soft Margins for AdaBoost, *Machine Learning*, vol. 42(3), pp. 287-320, March 2001. Also NeuroCOLT Technical report NCTR-1998-021

[58] Rubin, D.B., and Schafer, J.L., Efficiently Creating Multiple Imputation for Incomplete Multivariate Normal Data, *Proceedings of the Statistical Computing Section*, Alexandria: ASA, pp.83-8, 1990

[59] Rubin, D.B., Formalizing Subjective Notions about the Effect of Nonrespondents in Sample Surveys, *Journal of American Statistical Association*, vol.72, pp.538-543, 1977

[60] Rubin, D.B., Multiple Imputation After 18+ Years, *Journal of American Statistical Association*, vol.91, pp.473-489, 1996

[61] Rubin, D.B., *Multiple Imputations for Nonresponse in Surveys*, John Wiley and Sons: New York, 1987

[62] Rubin, D.B., Multiple Imputations in Sample Surveys, *Proceedings of the Survey Research Methods Section of the American Statistical Association*, pp.20-34, 1978

[63] Sande, G., *Hot-Deck Imputation Procedures, Incomplete Data in Sample Surveys*, Vol. 3. New York: Academic Press, 1983

[64] Sarle, W.S., Prediction with Missing Inputs, *Proceedings of Fourth Joint Conference on Information Sciences* (JCIS '98), vol. 2, pp.399-402, 1998

[65] Schapire, R.E., A Brief Introduction to Boosting, *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, pp. 1401-1406, 1999

[66] Schapire, R.E., and Singer, Y., Boostexter: A Boosting-Based System for Text Categorization, *Machine Learning*, vol. 39(2/3), pp. 135-168, 2000

[67] Schapire, R.E., and Singer, Y., Improved Boosting Algorithms Using Confidence-Rated Predictions, *Machine Learning*, vol. 37(3), pp. 297-336, December 1999. Also

Proceedings of the 14th Workshop on Computational Learning Theory 1998, pp. 80-91

[68] Shafer, J.L., *Analysis of Incomplete Multivariate Data*, Chapman and Hall, 1997

[69] Shafer, J.L., Multiple Imputations: A Primer, *Statistical Methods in Medical Research*, vol.8, pp.3-15, 1999

[70] Shannon, C.E., A Mathematical Theory of Communication, *Bell Systems Technical Journal*, vol.27, pp.379-423, 1948

[71] Singer, Y., Leveraged Vector Machines. Solla, S.A., Leen, T.K., and Muller, K.R., Editors, *Advances in Neural Information Processing Systems*, vol. 12, pp. 610-616, MIT Press, 2000

[72] Snedecor, G.W., and Cochran, W.G., *Statistical Methods*, Eighth Edition, Iowa State University Press, 1989

[73] Tresp, V., Neuneier, R., and Ahmad, S., Efficient Methods for Dealing with Missing Data in Supervised Learning, in Tesauro, G., Touretzky, D.S., and Leen, T.K. (Eds.), *Advances in Neural Information Processing Systems 7*, MIT Press, pp.689-696, 1995

[74] Vach, W., Missing Values: Statistical Theory and Computational Practice, In: Dirschedl, P., and Ostermann, R., (Eds.), *Computational Statistics*, Physica-Verlag, pp. 345-354, 1994

[75] Winter R., and Auerbach K., Contents Under Pressure, *Intelligent Enterprise*, available online at http://www.intelligententerprise.com/info_centers/scalability/showArticle.jhtml?articleID=18902161, May 2004

[76] Zhang, W., Association Based Multiple Imputation in Multivariate Datasets: A Summary, *Proceedings of the. 16th International Conference on Data Engineering (ICDE-2000)*, pp.310-311, 2000