RUTCOR
RESEARCH
REPORT

A New Imputation Method for Incomplete Binary Data

Mine Subasi[a] Ersoy Subasi[b]
Martin Anthony[c] P.L. Hammer[d]

RRR 15-2009, August 2009

RUTCOR
Rutgers Center for
Operations Research
Rutgers University
640 Bartholomew Road
Piscataway, New Jersey
08854-8003
Telephone: 732-445-3804
Telefax: 732-445-5472
Email: rrr@rutcor.rutgers.edu
http://rutcor.rutgers.edu/∼rrr

[a]RUTCOR, Rutgers Center for Operations Research, 640 Bartholomew Road Piscataway, NJ 08854-8003, USA. Email: msub@rutcor.rutgers.edu
[b]RUTCOR, Rutgers Center for Operations Research, 640 Bartholomew Road Piscataway, NJ 08854-8003, USA. Email: esub@rutcor.rutgers.edu
[c]Department of Mathematics, London School of Economics, Houghton Street, London WC2A 2AE, UK. Email: m.anthony@lse.ac.uk
[d]deceased.

Rutcor Research Report
RRR 15-2009, August 2009

A New Imputation Method for Incomplete Binary Data

Mine Subasi Ersoy Subasi Martin Anthony P.L. Hammer

Abstract. In data analysis problems where the data are represented by vectors of real numbers, it is often the case that some of the data points will have "missing values", meaning that one or more of the entries of the vector that describes the data point is not known. In this paper, we propose a new approach to the imputation of missing binary values. The technique we introduce employs a "similarity measure" introduced in [1]. We compare experimentally the performance of our technique with ones based on the usual Hamming distance measure and a more classical statistical technique.

1 Introduction

In practical machine learning or data analysis problems in which the data to be analyzed consists of vectors of real numbers, it is often the case that some of the data points will have "missing values", meaning that one or more of the entries of the vector that describes the data point is not known (or cannot be trusted). It is natural to try to "fill in" or *impute* these missing values so that one than has complete data to work from. This may be necessary, for instance, so that the data can be used to learn from using statistical or machine learning techniques. This is a classical statistical and machine learning problem and many techniques have been employed.

A variety of methods for imputing missing values have been proposed and analyzed (see Kalton and Kasprzyk [5]; Rubin [9]).

Some approached to handling missing data simply ignore or delete points that are incomplete. Classical approaches of this type are list-wise deletion (LD) and pairwise deletion (PD). Because of their simplicity, they are widely used (see, e.g., Roth [8]) and tend to be the default for most statistical packages. However, the application of these techniques may lead to a large loss of observations, which may result in data-sets that are too small if the fraction of missing values is high, and particularly if the original data-set is itself small.

Choosing the most appropriate method to handle missing data during analysis is one of the most challenging decisions confronting researchers. However, a naive or unprincipled imputation method may create more problems than it solves, distorting estimates, standard errors and hypothesis tests [6]. The most common data imputation techniques are mean imputation also referred to as unconditional mean imputation, regression imputation (RI) also referred to as conditional mean imputation, hot-deck imputation (HDI) and multiple imputation (MI). In most situations, simple techniques for handling missing data (such as complete case analysis methods LD and PD, overall MI, and the missing-indicator method) produce biased results as documented in [4, 7, 10, 12, 15]. A more sophisticated technique MI gives much better results [4, 7, 10, 12, 15].

MI [9] is a technique in which the missing values are filled several times, creating several completed data-sets for analysis. Each data-set is analyzed separately using techniques designed for complete data, and the results are then combined in such a way that the variability due to imputation may be incorporated. Rubin [10] gave a comprehensive treatment of MI and addressed potential uses of the technique primarily for large public-use data files from sample surveys and censuses. The technique is available in standard statistical packages such as SAS [11] and S-Plus. It has become increasingly attractive for researchers in the biomedical, behavioral, and social sciences where missing data is a common problem. These methods are documented in a recent book by Schafer [12] on incomplete multivariate data. MI is not the only principled method for handling missing values, nor is it necessarily the best for any given problem. In some cases, good estimates can be obtained through weighted estimation procedures. In fully parametric models, maximum-likelihood estimates can often be calculated directly from the incomplete data by specialized numerical methods, such as the Expectation-Maximization (EM) algorithm. The EM algorithm is an iterative procedure

in which it uses other variables to impute a value (Expectation), then checks whether that is the value most likely (Maximization). If not, it re-imputes a more likely value. This goes on until it reaches the most likely value. Those procedures may be somewhat more efficient than MI because they involve no simulation. EM Imputation is available in SAS, Stata, R, and SPSS Missing Values Analysis module.

In real-life applications missing data are a nuisance rather than the primary focus. In this case an easy solution with good properties can be preferable to one that is more efficient but problem-specific and complicated to implement.

In this paper, we propose a new approach to the imputation of missing binary values. The technique we introduce employs a "similarity measure" introduced in [1]. That measure has already proven to be of some application in classification problems [13]. Here, we use it to help indicate whether a missing value should be 0 or 1, and we compare experimentally the performance of our technique with ones based on the usual Hamming distance measure and a more classical statistical technique.

The framework used here requires data to be represented by binary vectors. However, in many applications, the raw data that we work with in a particular situation might be more naturally encoded as a real-valued vector. In such cases, the data may be transformed into binary data through a process known as *binarization* (see [2] for example). The transformed data-set may then be simplified or cleaned in a variety of ways, by the removal of repeated points, for instance, and the deletion of attributes (or co-ordinates) found to be statistically insignificant in determining the classification.

Section 2 provides details of the Boolean similarity measure that is at the core of our technique and describes the imputation method that derives from this measure. Section 3 describes the experiments we performed in order to test this method, and the results are reported in Section 4.

2 A measure of similarity and its application to missing values

In [1], a way of measuring the similarity $s(x, A)$ of a Boolean vector x to a set A of such vectors is proposed. The measure can be described in two ways: either in terms of Boolean functions or, in a combinatorial way, in terms of substrings.

2.1 A Boolean function description

Any Boolean function $f : \{0,1\}^n \to \{0,1\}$ can be expressed by a *disjunctive normal formula* (or DNF), using *literals* $u_1, u_2, \ldots, u_n, \bar{u}_1, \ldots, \bar{u}_n$, where the $\bar{u}_i$ are known as *negated literals*. A disjunctive normal formula is one of the form

$$T_1 \vee T_2 \vee \cdots \vee T_k,$$

where each T_l is a *term* of the form

$$T_l = \left(\bigwedge_{i \in P} u_i \right) \bigwedge \left(\bigwedge_{j \in N} \bar{u}_j \right),$$

for some disjoint subsets P, N of $\{1, 2, \ldots, n\}$. A Boolean function is said to be a k-DNF if it has a disjunctive normal formula in which, for each term, the number of literals ($|P \cup N|$) is at most k. Such a function is said to be an l-term k-DNF if, additionally, it has a k-DNF formula in which the number of terms is at most l. For two Boolean functions f and g, we write $f \leq g$ if $f(x) \leq g(x)$ for all x; that is, if $f(x) = 1$ implies $g(x) = 1$. Similarly, for two Boolean formulae ϕ, ψ, we shall write $\phi \leq \psi$ if, when f and g are the functions represented by ϕ and ψ, then $f \leq g$. A term T of a DNF is said to *absorb* another term T' if $T' \leq T$. A term T is an *implicant* of f if $T \leq f$; in other words, if T true implies f true. The terms in any DNF representation of a function f are implicants of f. The most important type of implicants are the *prime implicants*. These are implicants with the additional property that there is no other implicant of f absorbing T. Thus, a term is a prime implicant of f if it is an implicant, and if the deletion of any literal from T results in a non-implicant T' of f (meaning that there is some x such that $T'(x) = 1$ but $f(x) = 0$). If we form the disjunction of all prime implicants of f, we have a DNF representation of f.

We now give the definition of *similarity* that is the focus of this paper.

Definition 1. *Suppose that A is a given subset of $\{0,1\}^n$. Let F be the Boolean function whose value is 1 in every point not in A, and 0 in every point of A, so F is the indicator function of $\bar{A}$, the complement of A. Let G_k be the disjunction of all prime implicants of length k of the function F. Then we define the similarity of $x \in \{0,1\}^n$ to A, denoted $s(x, A)$, to be the smallest k for which $G_k(x) = 1$.*

This is a slightly different description from that given in [1], but it is equivalent to the formulation given there.

2.2 A combinatorial description

There is another useful way of describing the similarity measure. Suppose $x \in \{0,1\}^n$, $I \subseteq [n] = \{1, 2, \ldots, n\}$, and $|I| = k$. Then the projection of x onto I is the k-vector obtained from x by considering only the coordinates in I. For example, if $n = 5$, $I = \{2, 4\}$ and $x = 01001$ then $x|_I = 10$.

By a *positional substring* of $x \in \{0,1\}^n$, we mean a pair (z, I) where $z = x|_I$. The key point here is that the coordinates in I are specified: we will want, as part of our later definitions, to indicate that two vectors x and y have the same entries *in exactly the same places*, as specified by some $I \subseteq [n]$. For instance, although both $x = 10101$ and $y = 01010$ have substrings equal to 00, there is no I such that $x|_I = y|_I = 00$.

We can now give an equivalent definition of similarity, from [1].

Definition 2. *For $A \subseteq \{0,1\}^n$ and $x \in \{0,1\}^n$, the* similarity *of x to A, $s(x,A)$, is defined to be the largest s such that every positional substring (x,I) of length s appears also as a positional substring (y,I) of some $y \in A$. That is,*

$$s(x,A) = \max\{s : \forall I \subseteq [n], |I| \leq s, \exists y \in A, y|_I = x|_I\}.$$

Here $x|_I$ denotes the projection of x onto the coordinates indicated by I.

Equivalently, if r is the smallest length of a positional substring possessed by x that does *not* appear (in the same positions) anywhere in A, then $s(x,A) = r - 1$.

Notice that $s(x,A)$ is a measure of how similar x is to a *set* of vectors. It is not a metric or distance function. It can immediately be seen, indeed, that if A consists solely of one vector y, not equal to x, then $s(x,A) = 0$, since there must be some coordinate on which x and y differ (and hence a positional substring of length 1 of x that is absent from A).

Informally, then, the similarity of x to A is low if x has a short positional substring absent from A; and the similarity is high if all positional substrings of x of a fairly large length can be found in the same positions in some member y of A.

2.3 Example

Example Suppose the set A consists of the following 10 points of $\{0,1\}^5$.

$$\begin{array}{ccccc} 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \end{array}$$

Note, first, that no x can have $s(x,A) = 0$, since this could only happen if, on one of the five coordinates, all elements of A had a fixed value, either 0 or 1. Consider any x of the form $x = 01x_3x_4x_5$. Since there is no $y \in A$ with $y|_{\{1,2\}} = x|_{\{1,2\}} = 01$, we have $s(x,A) = 1$. Consider, however, $x = 10101$. For this x, we have $s(x,A) = 3$, because all (positional) substrings of x of length 3 belong to A, but there is no $y \in A$ such that $y|_{\{1,2,4,5\}} = x|_{\{1,2,4,5\}} = 1001$. Suppose now that $x = 00001$. Then, since all (positional) substrings of x of length 2 appear in A, $s(x,A) \geq 2$. However, there are substrings of length 3 missing from A: for example, there is no $y \in A$ with $y|_{\{1,3,4\}} = x|_{\{1,3,4\}} = 000$. So $s(x,A) = 2$.

2.4 A new technique for imputing missing values

In general terms, a natural approach to determining a missing value in a data-point is to find the value which will make the data-point appear to be most in line with the rest of the data-set. For example, suppose the first value of a data-point is missing. Then we may think it sensible to replace it by the average of all the known first values of all the other points in the data set. This is indeed a commonly-used technique. (When the data consists of binary vectors, then the counterpart to this technique is the MAJORITY method, in which the missing entry is replaced by the value, 0 or 1, that is most commonly taken in that same position in the rest of the data.)

The general approach we propose here applies to binary data, and is as follows: suppose $x \in \{0,1\}^n$ belongs to a data-set A and that x has one or more missing values. Then the values we impute to those missing values are those that maximize the similarity of the resulting vector to the subset A^* of the data-set consisting of all the data-points with no missing values. That is, we impute the missing values in such a way that the resulting vector $\hat{x}$ minimizes $s(\hat{x}, A^*)$. If there is more than one assignment of values achieving the maximum similarity, the values can be assigned randomly among these.

For example, suppose that $x \in \{0,1\}^n$ is missing one value, in its first entry. Let us indicate this by writing $x = \mathbf{?}x'$, where $x' \in \{0,1\}^{n-1}$. (The '**?**' symbol denotes the missing value.) If $s(1x', A^*) > s(0x', A^*)$ then we will take $\mathbf{?} = 1$, so that x, once the missing value is imputed, becomes $1x'$. If $s(1x', A^*) < s(0x', A^*)$, then we will take $\mathbf{?} = 0$. If $s(1x', A^*) = s(0x', A^*)$, then we will take $\mathbf{?} = 1$ or 0, randomly (with equal probability).

3 Experiments

3.1 An overview of the experiments

To test the method, we conducted experiments based on real data-sets (binarized if necessary) in which single entries of some data-points are assumed to be missing. We compared the performance of the method with that of other methods. For each data-set D, and each of the imputation methods considered, we randomly selected 10% of the data-set, giving a subset D' of D.

Then, for each $x \in D'$ and each $i \in \{1, 2, \ldots, n\}$, supposing the ith entry of x (but no others) to be missing, we used the imputation technique to determine what that value should be, taking the data-set to be $(D \setminus D') \cup \{x\}$. This involves mn applications of the imputation technique, where n is the number of attributes of the data (that is, its dimension) and m the size of the selected set D'. In each case, we knew what the true value should have been, so we can know whether the method has worked correctly or not. So, explicitly, for the similarity-based method, for $x \in D'$, let us denote by $x^{(i)}$ observation x with its ith entry assumed to be missing. Let $x_1^{(i)}$ be x with ith entry taking value 1 (so this is the binary vector that agrees with x in all entries except possibly entry i, which is 1) and define $x_0^{(i)}$ similarly. Then, we determine the similarities $s_1 = s\left(x_1^{(i)}, D \setminus D'\right)$ and $s_0 = s\left(x_0^{(i)}, D \setminus D'\right)$.

If $s_1 > s_0$, then we impute value 1 for the ith entry of x; if $s_1 < s_0$, then we impute value 0; and if $s_1 = s_0$, then the method fails to determine uniquely a value to impute (and in this case, one could instead simply choose randomly between 0 and 1).

For each of the methods we consider, and on each data-set, we determined three numbers that describe the performance of the method:

- the percentage of times the method determined (uniquely and unambiguously) the correct value;
- the percentage of times it did not determine uniquely a value to impute;
- the percentage of times the method did uniquely determine a value, but the value determined was incorrect.

We call the third of these three statistics the *error rate* of the method in this experiment.

In the second of the cases above, in which the value is not determined, one could chosen a value randomly. Of course, the resulting imputation will be incorrect some of the time, so it could be argued that the true error rate should take this into account. But what we want to assess is the extent to which the method fails when it *definitively* (unambiguously) makes a decision as to what the missing value should be. When the decision is ambiguous, we could, instead of choosing randomly, simply declare that the method has not determined the missing value, or we could invoke some other method. In applications where it is acceptable to have some points remaining incomplete, or where the penalty for imputing wrong values is severe, the appropriate error measure is then the one we propose.

In addition to the similarity-based method, which we'll denote SIM, we considered several others. One, which we call the Hamming method, denoted HAMM, uses Hamming distance as a guide to how similar a point is to the others. This method acts as follows: suppose $x \in \{0,1\}^n$ belongs to a data-set D and that x has one or more missing values. Then the values we impute to those missing values are those such that the resulting vector $\hat{x}$ will *minimize* the Hamming distance

$$d_H(\hat{x}, D^*) = \min\{\sum_{i=1}^{n} |\hat{x}_i - y_i| : y \in D^*\}$$

of $\hat{x}$ to the subset D^* of the data-set consisting of all the data-points with no missing values. (If there is more than one such possible $\hat{x}$ then one is chosen randomly from among these.)

So, in the context of our single-missing-value experiments, and with the notation as above, the Hamming method acts as follows: we determine the Hamming distances $d_1 = d_H\left(x_1^{(i)}, D \setminus D'\right)$ and $d_0 = d_H\left(x_0^{(i)}, D \setminus D'\right)$. If $d_1 < d_0$, then we impute value 1 for the ith entry of x; if $d_1 > d_0$, then we impute value 0; and if $d_1 = d_0$, then we choose randomly between 0 and 1.

Two methods can be derived by using the similarity measure and Hamming distance together.

Dataset	# of observations		# of attributes		After preprocess		
					# of observations		# of binary attributes
	Positive	Negative	Numeric	Nominal	Positive	Negative	
Cleveland Heart Disease	139	164	10	3	137	158	63
Pima Indian Diabetes	130	262	8	0	130	262	47
German credit	700	300	7	13	697	300	66
Hepatitis	123	32	6	13	92	19	28
Ionosphere	225	126	34	0	216	125	49
Mushroom	3916	4208	0	22	2188	2047	50
Tic-Tac-Toe	626	332	0	9	626	332	27
Voting	267	168	16	0	96	64	16
Wisconsin Breast Cancer	458	241	9	0	203	182	48

Table 1: Nine data-sets

First, SIM & HAMM acts as follows: first, SIM is applied. If that does not return a uniquely determined (that is, non-random) value for the missing value, then HAMM is then applied to determine the missing value. This subsequent application of HAMM may, of course, still not uniquely determine a value, in which case HAMM chooses randomly as its definition requires.

Secondly, HAMM & SIM first applies HAMM and, if that does not uniquely determine the missing value, then SIM is employed.

Another method is to apply MI techniques [10] that are implemented in PROC MI [16] using SAS [11]. This method always determines uniquely a value for the missing value.

Finally, we also used the MAJORITY method in which the missing value imputed into $x^{(i)}$ is 1 (respectively, 0) if a majority of the elements of $D \setminus D'$ have a 1 (respectively, 0) is position i, and is chosen to be 0 or 1 at random if each occurs equally often.

3.2 The data-sets

In our experiments we used the following nine data-sets, taken from the UCI Machine Learning Repository [14]. The data-sets were pre-processed in several ways before we ran our experiments. First, any observations in the data-set that had any missing attribute values were deleted. Next, the data-sets were binarized, according to the method described in [2], so that any numerical or nominal attribute values were changed to binary values. Next, techniques from [3] were used to determine that some attributes (of the binarized data) could be deemed irrelevant and therefore deleted. (Set covering was used to find a small "support set".) The binarized data was then projected onto the remaining binary attributes. If this process resulted in any repetitions, these were deleted, and if any of the processed observations appeared once with each class label, all its occurrences were deleted. After pre-processing in this manner, the data-sets consisted of binary vectors, generally in a higher-dimensional space than the original data. Table 1 describes the characteristics of the data-sets before and after this pre-processing.

Dataset	correct determination	non-determination	error rate
Breast Cancer	90%	9%	1%
Diabetes	76%	23%	1%
German Credit	96%	4%	0.21%
Heart Disease	70%	28%	1%
Hepatitis	39%	55%	6%
Ionosphere	58%	41%	2%
Mushroom	99%	1%	0.01%
Tic-Tac-Toe	77%	15%	8%
Voting	48%	49%	3%

Table 2: Imputation using similarity with SIM

Dataset	correct determination	non-determination	error rate
Breast Cancer	81%	10%	8%
Diabetes	76%	9%	15%
German Credit	78%	9%	13%
Heart Disease	84%	6%	10%
Hepatitis	61%	16%	23%
Ionosphere	82%	5%	13%
Mushroom	82%	17%	1%
Tic-Tac-Toe	55%	15%	30%
Voting	55%	36%	9%

Table 3: Imputation using Hamming distance with HAMM

4 Experimental results

In this section we present experimental results obtained by the use of imputation techniques described in Section 3.1. In each table we show the percentages of times the method determined (uniquely and unambiguously) the correct value (*correct determination*); the percentage of times it did not determine uniquely what the value should be (*non-determination*) and the percentage of times the method did uniquely determine a value, but the value determined was incorrect (*incorrect determination; that is, error*).

For all data-sets Tables 2–7 show the rates of correct determination, non-determination and errors (incorrect determination) obtained by different imputation methods. Table 8 shows the average *determination rate* (by which we mean the percentage of times the missing value was determined, either correctly or incorrectly) over all nine data-sets, and the *correct determination rate* (by which we mean the proportion of determinations that are correct) for all imputation methods discussed in Section 3.1.

Dataset	correct determination	non-determination	error rate
Breast Cancer	95%	2%	3%
Diabetes	86%	4%	10%
German Credit	99%	0%	1%
Heart Disease	90%	3%	7%
Hepatitis	66%	13%	21%
Ionosphere	86%	4%	10%
Mushroom	99%	1%	0.04%
Tic-Tac-Toe	80%	2%	18%
Voting	71%	19%	10%

Table 4: Imputation Combining similarity and Hamming distance: SIM & HAMM

Dataset	correct determination	non-determination	error rate
Breast Cancer	90%	2%	9%
Diabetes	81%	4%	15%
German Credit	86%	0%	13%
Heart Disease	87%	3%	10%
Hepatitis	63%	13%	24%
Ionosphere	83%	4%	13%
Mushroom	99%	1%	1%
Tic-Tac-Toe	67%	2%	31%
Voting	69%	19%	11%

Table 5: Imputation combining similarity and Hamming distance: HAMM & SIM

Dataset	correct determination	non-determination	error rate
Breast Cancer	97%	0%	3%
Diabetes	88%	0%	12%
German Credit	94%	0%	6%
Heart Disease	94%	0%	6%
Hepatitis	69%	0%	31%
Ionosphere	87%	0%	13%
Mushroom	96%	0%	4%
Tic-Tac-Toe	100%	0%	0%
Voting	73%	0%	28%

Table 6: Imputation using SAS

Dataset	correct determination	non-determination	error rate
Breast Cancer	62%	2%	36%
Diabetes	70%	0%	30%
German Credit	71%	0%	29%
Heart Disease	76%	0%	24%
Hepatitis	71%	0%	29%
Ionosphere	65%	0%	35%
Mushroom	71%	0%	29%
Tic-Tac-Toe	71%	0%	29%
Voting	61%	0%	39%

Table 7: Imputation using MAJORITY

	Determination rate	Correct determination rate
SIM	75%	97%
SIM & HAMM	95%	91%
SAS	100%	89%
HAMM & SIM	95%	85%
HAMM	86%	84%
MAJORITY	100%	69%

Table 8: Overall Results

It appears from the results of the experiments that the similarity-based method achieves, on average, a lower error rate than the other methods. What this means is that when it is used to determine a missing value, if it does so uniquely, then it appears to perform relatively well. However, as Table 8 makes clear, the method also generally has a lower determination rate than the other methods. So there is, in a sense, a trade-off between determination rate and error rate. If it is important to be sure of imputed values, and also acceptable to have some missing values as yet undetermined, then the similarity-based approach looks like a useful one. In the experiments, the HAMM method achieved a higher determination rate, but a lower rate of correct determination, than the SIM method. When the SIM and HAMM methods are used in combination, a higher still determination rate is achieved, and the rate of correct determination appears to be as good as (and possibly better) than other methods (and better than the HAMM method used on its own).

Acknowledgements

Martin Anthony's work is supported in part by the IST Programme of the European Community, under the PASCAL2 Network of Excellence. Peter L. Hammer's work was partially supported by NSF Grant NSF-IIS-0312953 and NIH Grants NIH-002748-001 and NIH-HL-072771-01.

References

[1] Martin Anthony and Peter L. Hammer. A Boolean measure of similarity. *Discrete Applied Mathematics* 154(16): 2242–2246, 2006. (Also, RUTCOR Research Report RRR-27-2004, RUTCOR, Rutgers Center for Operations Research, Rutgers University, New Jersey, August 2004.)

[2] Endre Boros, Peter L. Hammer, Toshihide Ibaraki and Alexander Kogan. Logical analysis of numerical data. *Mathematical Programming* 79, 1997: 163–190.

[3] Boros E., Peter L. Hammer, Toshihide Ibaraki, Alexander Kogan, Eddy Mayoraz and Ilya Muchnik. An Implementation of Logical Analysis of Data. *IEEE Trans. on Knowledge and Data Engineering* 12-1 (2000): 292–306.

[4] Greenland S., Finkle WD. A critical look at methods for handling missing covariates in epidemiologic regression analyses. Am J Epidemiol 1995, 142:1255–64.

[5] Kalton, G., and Kasprzyk, D., Imputing for Missing Survey Responses, Proceedings of the Survey Research Methods Section, Washington, D.C.: American Statistical Association (1982), 22–31.

[6] Little, R.J.A. and Rubin, D.B. (1987) Statistical Analysis with Missing Data. J. Wiley & Sons, New York.

[7] Little R.A., Regression with missing Xs; a review. J Am Stat Assoc 1992, 87:1227–37.

[8] Roth, P., Missing data: a conceptual review for applied psychologists. Personnel Psychology (1994) 47, 537-560.

[9] Rubin, D.B., Multiple imputations in sample surveys - a phenomenological Bayesian approach to nonresponse, Amer. Statist. Assoc. 1978 Proc. Section on Survey Res. Methods (1978) 20–34.

[10] Rubin, D.B. (1987) Multiple Imputation for Nonresponse in Surveys. J. Wiley & Sons, New York.

[11] SAS. SAS/STAT Users Guide. Version 8. SAS Institute, Inc: Cary, NC, 1999.

[12] Schafer JL. Analysis of incomplete multivariate data. London: Chapman & Hall/CRC Press; 1997.

[13] Subasi M., Subasi S., Anthony M., and P.L. Hammer, Using a similarity measure for credible classification. Discrete Applied Mathematics (2009) 157 (5), 1104–1112.

[14] University of California at Irvine Machine Learning Repository.

`http://www.ics.uci.edu/ mlearn/MLRepository.html`

[15] Vach W. Logistic regression with missing values in the covariates. New York: Springer, 1994.

[16] Yuan, Y.C. Multiple Imputation for Missing Data: Concepts and New Development. SUGI Proceedings 2000, 267–25.