ITC 4/49

Information Technology and Control
Vol. 49 / No. 4 / 2020
pp. 541-555
DOI 10.5755/j01.itc.49.4.27386

A New Approach for Missing Data Imputation in Big Data Interface

Received 2020/07/24 | Accepted after revision 2020/11/04

http://dx.doi.org/10.5755/j01.itc.49.4.27386

HOW TO CITE: Wang, C., Shakhovska, N., Sachenko, A., Komar, M. (2020). A New Approach for Missing Data Imputation in Big Data Interface. *Information Technology and Control*, 49(4), 541-555. https://doi.org/10.5755/j01.itc.49.4.27386

A New Approach for Missing Data Imputation in Big Data Interface

Chunzhi Wang
School of Computer Science, Hubei University of Technology, Nanli Road, 28, Hong-shan District, Wuhan 430068, China; e-mail: chunzhi2020@163.com

Nataliya Shakhovska
Artificial intelligence department, Lviv Polytechnic National University, 12 Bandera str, Lviv, 79013, Ukraine; e-mail: nataliya.b.shakhovska@lpnu.ua

Anatoliy Sachenko
Department of Informatics, Kazimierz Pulaski University of Technology and Humanities in Radom, 26600 Radom, 29 Malczewski Str, Radom, Poland
Research Institute for Intelligent Computer Systems Ternopil National Economic University, 3 Peremoga Square, Ternopil 46020 Ukraine; e-mails: as@tneu.edu.ua, mko@tneu.edu.ua

Myroslav Komar
Research Institute for Intelligent Computer Systems Ternopil National Economic University, 3 Peremoga Square, Ternopil 46020 Ukraine; e-mails: as@tneu.edu.ua, mko@tneu.edu.ua

Corresponding author: nataliya.b.shakhovska@lpnu.ua

The three-stage approach for missing data imputation in Big data interface is proposed in the paper. The first stage includes designing the Big data model in the task of missing data recovery, which enables to process the structured and semistructured data. The next stage is developing the method of missing data recovery based on functional dependencies and association rules. Estimating the algorithm complexity for missing data recovery is provided at the last stage. The proposed method of missing data recovery creates additional data values using a based domain and functional dependencies and adds these values in available training data. Performing the analysis of data different types is possible too. The correctness of the imputed values is verified on the classifier built on the original dataset. The proposed method performs 12% better than the Expectation-Maximization (EM) and Random Forest (RF) methods for 30% missing data and enables the parallel execution in distributed databases. The acceleration for m=41 attributes is 12.5 times larger for 20 servers (processors) compared to the non-parallel mode.

KEYWORDS: Big data, semistructured data, missing data recovery, functional dependency, association rule.

1. Introduction

Missing values cause severe issues in data analysis. The various algorithms for data analysis require a matrix of indicators without gaps. The missing value imputation is an important thing for IoT solution and Big data analysis and preprocessing, too. For example, for medical decision support we need to determine the physical and psycho-emotional characteristics of the patient as well as the environmental indicators (indoor and outdoor air temperature, humidity, pressure, etc.). Other example is a smart city platform that consists of information about transport location, CO_2 level, energy consuming as well as information about health parameters of the citizens [27]. This means that streaming data are analyzed. Moreover, a tremendous growth of sensors and their acquisition, collection and storing arise the new missing data issues. Besides missing values, like any method of data analysis, require an understanding of their nature. The key issue in the analysis of incomplete data is the mechanism of gaps in them.

The goal of the paper is to improve the recovery of missing data using the Probabilistic Production Dependency.

The main contribution consists of the following:

- a Big data model for recovery of missing data which enables to process the semistructured data;
- a method for missing data recovery based on functional dependencies and association rules;
- a two-stage technique of estimating the algorithm complexity for missing data recovery.

The novelty of the proposed method for imputation of missing data is to process the structured and semistructured data based on the hierarchy of the objects as well as the ensemble of functional dependencies and association rules development. This issue is very important for Big data interfaces because the most of information is available in semistructured form. The proposed algorithm creates additional data values using domain based and functional dependencies based on multiple imputation methods and adds these values in available training data. The correctness of the imputed values is verified on the classifier built on the original dataset. A comparison of the methods of analysis and modeling of statistical processes by qualitative criteria is made.

The paper consists of five sections. The rest of the paper is structured as follows. Section 2 describes a state-of-the-arts, Section 3 presents the proposed approach for recovering the missed data, Section 4 covers the experimentation and results. The last Section 5 summarizes the obtained outcomes.

2. Related Work

In this section, we are focusing on recent developments and important information about recovery of missing data.

There are three types of gap generation mechanisms: missing completely at random (MCAR), missing at random (MAR) and missing not at random (MNAR) [1]. For MCAR, the probability of observing data gaps does not depend on either observable or non-observable metrics. For MAR, the probability of observing data gaps depends on the observed metrics: the probability that the value of X_i is omitted is not related to X_i itself, but it depends on other variables in the analyzed table. For MNAR, the probability of observing data gaps depends on the values of the missing variables that are not available for analysis: the probability that the value of X_i is omitted is associated with X_i itself.

A common issue with MCAR is that the data under this scheme at the beginning of the analysis may actually be different because of unknown or unaccepted factors. On this basis, incomplete data often have a non-MCAR scheme, but they may be assigned to the scheme with occasional MAR omissions in the event that incomplete data may occur depending on the known indicators. If incomplete data cannot be attributed to either MCAR or MAR, they are classified as MNAR. This means that modeling results based on such data will have biased estimates if the missing value model is unknown or disregarded.

The main disadvantage of MCAR, MAR and MNAR methods is absence of binding with either data source or data structure. It means the impossibility to predict the place of omitted data while such type of prediction is very important for streaming data. That is why these methods are suitable for missing data recovery in large homogenous datasets only [11].

In general, there is no way to determine the omission pattern if the values of the omitted data themselves are unknown. Because of this fact, researchers usually claim that there is a mechanism based on their own experience and the nature of the application data [4].

The following methods are based on the model of missing data or used machine learning for omission estimation.

Methods based on the skip generation model. This group includes methods of analysis (usually frequency or Bayesian) based on probability estimation. Their application requires a preliminary assessment of the mechanism for gaps generation. If the MCAR or MAR are assumed, there is no need to take into account the omission model. Otherwise, the omission generation model should be included in the analysis process in the presence of the MNAR scheme. Unlike fill-in approaches, these methods do not restore any visible values. On the contrary, the data are processed as if they were fully complete, and the corresponding parameters or statistics are estimated in the same way [5].

Methods of imputation (restoration, filling). Imputation is a procedure for estimating the unknown values based on the available data, which leads to a complete set of data with some plausible estimates at the point of omission.

The imputation method group contains two types of methods that are related to the use of some models, namely, non-model or model-based approaches. In terms of the number of values that result from the use of imputation methods, there are approaches based on single imputation methods and multiple imputation methods [6]. Single-fill algorithms give a single complete set of data, where each space is replaced by a value. The advantages of this approach are using the full data analysis methods in the subsequent processing steps and the reduced work for filling each gap. Algorithms of multiple filling generate several complete sets of data, which are analyzed separately and the obtained results are combined. That enables using the full data analysis methods in the subsequent data processing steps and provides improved estimates as well as minimizes standard errors. However, these methods require many resources to create more data sets and need more time to perform analysis as well as more memory to store results [9].

Let us consider the first group of single-fill methods on a non-model and model basis. *Gaps* on a non-model basis implement a simple procedure of substituting the certain values instead of missing ones and, like deletion methods [10], belong to the group of traditional methods for processing the incomplete data.

Mean substitution (MS). This method enables solving the problem of data incompleteness replacing each missing by an average variable. Types of MS are the substitution of the median, mode, mean of data subgroup [10], the value with the highest frequency (most common value), in some cases, replacing the minimum / maximum value. This method can lead to many undesirable results [13], such as underestimation of the real variance, negative bias of correlations and incorrect representation of the general population.

Simple *hot-deck imputation* replaces each missing value with a random one taken from an existing set of data [16]. Its significant drawback is the distortion of correlations and covariates.

Cold-deck imputation (CD) implements the replacement of each pass by some constant value from an external source [23]. A particular case of CD is zero substitution. Like the consistent and occasional hot substitution, it has the same disadvantages as the simple hot substitution.

The advantage of *substitution-based approaches* is that they give an internally consistent set of values, but cannot define the relationships between the variables. Model-based gap filling methods estimate the values of model parameters that are consistently used to impute the gaps. Those methods take into account the relationships between variables, but they are rather complex because they cause errors when all parameters are evaluated simultaneously.

Regression estimation (RE) involves replacing data gaps with predicted values derived from a regression equation constructed from a complete set of data. Similar to other filling methods, the RE is sampled as the initial volume. The drawbacks of regression filling include the following: the need to identify regression models precisely, exaggeration of correlation and covariance, a probability of going the predicted values beyond the logical series, and the need for large amounts of data to obtain consistent estimates.

Maximum likelihood (ML) method outputs estimated parameters in such a way that the probability of data reproduction based on known values is maximized. The ML method is considered as one of the approaches for processing the incomplete data [18], and it is widely recommended for use since it does not lead to bias in the presence of missing schemes MCAR and MAR values [17]. However, the ML method does not solve this issue in the presence of MNARs, although the magnitude of such bias tends to be much smaller than traditional approaches.

The expectation maximization (EM) method is based on a general iterative algorithm that implements the sequential filling of missing values by their estimates and maximizes the conditional expectation of the log likelihood for complete data to the convergence process [17]. The main drawback of ML and EM methods is their iteration property and a high time complexity afterwards, especially for Big data processing.

The k-nearest neighbor (kNN) method is based on the determination of the nearest neighbor for each missed value using a specific metric [12]. After finding the k-nearest neighbors per each pass, it is replaced by the average value of characteristics for its neighbors containing complete data. A special case of kNN method is the weighted kNN method [3]. The greatest difficulty in its application is to determine an adequate degree of closeness.

Support vector machine (SVM) [2] algorithm avoids probability estimation on data which are stable. The kernel choosing has influence on the quality of the missing data imputation. That is why the implementation of this algorithm depends on the dataset and its parameters.

Random forest (RF) method [26] is based on the construction of a classifier formed by a group of decision trees, a training set of complete data, and the prediction of missing values in a test set. In the first step, all gaps in the test set are filled with an average based on the observed data. Next, the proximity matrix is calculated for the first iteration data set. Average weights computed on the proximity matrix for the first iteration are used to replace the gaps at the next iteration and so on until the stop criterion is reached [19]. Although this method is iterative, it enables parallelizing the processing by reducing the time complexity.

Association rules (AR) method involves constructing associative rules on the excessiveness of available data [20, 22]. Firstly, all received rules are sorted by some parameter (for example, authenticity). Then, the data set is searched per each omission and for a corresponding rule, which does not contradict other values that are in the recoverable row. The AR algorithm has a several modifications. For example, one of them, the FP-algorithm can run in parallel efficiently, which is why it can be used for Big data preprocessing. However, the time complexity of this method has to be improved [25].

Multilayer perceptron (MLP) based method uses a trained network to estimate the missing values. As inputs are accepted non-empty data sets, outputs are incomplete values of variables that need to be defined [7, 17, 21]. Filling in the missing values with MLP consists of three steps:

1. forming a complete (training) and incomplete (test) set;
2. construction of MLP on the training set, where values of the variable are the values of gaps, and the input values, are respectively, the filled values of the variable;
3. predicting the unknown values for each incomplete data structure using a trained network. The drawback of this approach is binding to a set of variables that contain missing values, so one needs to build a separate MLP model per each combination.

Self-organizing map (SOM) method [8] allows training a network based on incomplete input set. It is running due to the ability of the algorithm to ignore the missing values and calculate the distance between the current observation and the node and use the trained map to evaluate the values. The disadvantage of SOM is the same as of that MLP.

The biggest problem of the methods analyzed above is the processing of structured data only while the semistructured and unstructured data are dominating in real situations [15, 24]. In addition, it is very important to analyze the hidden dependencies in the dataset as well as to take into account the character of dataset and predict the missing data appearance per each data source separately. That is why the paradigm of Big data is used for preprocessing and processing information from various sources consisting the continuous numeric data and categorical data. Nevertheless, the nature of missing data in different data sources is different too. Hence, the main idea is to analyze data from different sources based on the specificity and nature of these sources.

3. Materials and Methods

To solve the issue indicated above, authors propose a three-stage approach, which includes: (i) designing the Big data model in the task of missing data recovery, (ii) developing the method of missing data recovery based on functional dependencies and association rules, and (iii) estimating the algorithm complexity for missing data recovery.

3.1. The Big Data Model in the Task of Missing Data Recovery

We propose to analyze the structure of datasets to find the uncertainty types and to create identity function (mediator) to a general schema of Big data based on all data sources [24]. After that, the general schema can be used for missing data recovery directly in the data source.

First, the Big data schema should be defined.

The Big data schema Bd is the finite set of attributes with exact values $\{A_1, A_2, ..., A_n\}$, set of attributes with indefinite, inexact or missing values $\{A\_unk_1, A\_unk_2, A\_unk_p\}$ and the set of values of membership functions for inexact attributes $\{Unk_1, Unk_2, \ldots, Unk_m\}$. In addition, the catalog of the attributes (features) with schema Cg and synonym dictionary with schema *Dic* should be used:

$$Bd =< \{A_1, A_2, ..., A_n\}, \{A\_unk_1, A\_unk_2, A\_unk_p\}, \\ , \{Unk_1, Unk_2, \ldots, Unk_m\}, Dic, Cg > . \quad (1)$$

The attributes of the ***A_unk*** set are considered indeterminate, and the level of confidence in them is stored in the attributes of the set ***Unk*** .

A binary relation *Rel* is used to show the relationships between the attributes of the and ***Unk*** sets, the values of which are determined based on the sample view of the source and in the Cg data directory:

$$Rel = \left| rel_{ij} \cdot \sigma_{\arg(i)}(Cg) \right|, \ \forall i = \overline{1, p}, \forall j = \overline{1, m} \\ rel_{ij} = \begin{cases} 1, \ Unk_j \Leftrightarrow A_{unk_i} \wedge \sigma_{\arg(j)}(Dic) \\ 0, \ otherwise \end{cases} \quad (2)$$

The sum of the binary rows is 1, since we assume that the degree of trust in an attribute will not be indicated by two or more attributes from the set ***Unk***:

$$\forall i = \overline{1, p}, \sum_{1}^{n} meta_{ij} = 1 . \quad (3)$$

The *Rel* relationship will allow us to model any type of uncertainty without extending attribute domains.

Big data instance *bd* is an information description of an object t of a data source S, represented as a set (tuple) of values of characteristics (attributes), a subset of attribute values of which contains data about the object, data source and synonymous names of the object, and these data may be missing, incomplete, fuzzy, or undetermined [23]. That is, the object being modeled on the data source by this tuple exists, but some information about it is missing, fuzzy, incomplete, undetermined, and so on.

Here are examples of consolidated data tuples for different types of information resources.

1 Relational database - in this case an extended relational tuple is used t_{rel}:

$$bd = t_{rel} \cup Unk, \\ t_{rel} = \{a_1, ..., a_n\} \cup \{a\_unk_1, ..., a\_unk_m\}, \quad (4)$$

where $\{a_1,...,a_n\}$ are the value of exact attributes, $\{a\_unk_1,...,a\_unk_m\}$ are the value of attributes with uncertainty.

2 Data Warehouse combines fact and dimension data. A set of measurement values and fact characteristics is presented as a tuple t_{dw}:

$$\begin{aligned} &bd = t_{dw} \cup Unk, \\ &t_{dw} = \{a_1,...,a_n\} \cup \{a\_unk_1,...,a\_unk_m\} \cup \\ &\cup\{a_{rf1},...,a_{rfi}\} \cup \{a\_unk_{11},...,a\_unk_{1m}\} \cup ... \\ &... \cup \{a\_unk_{k1},...,a\_unk_{ks}\} \cup ... \\ &... \cup \{a\_unk_{rf1},...,a\_unk_{rft}\} \cup, \end{aligned} \tag{5}$$

where $a_{i,j}$ is the value of exact characteristic j in the dimension i, a_{rf1} is the value of the characteristic j of fact table, $a\_unk_{i,j}$ is the value of attribute with uncertainty j from dimension i, a_unk_{rfj} is value of attribute with uncertainty j from the fact table.

3 Semi-structured text describes the values of the nodes of the semantic networks and the degree of affiliation of these values to the objects whose names are described in the synonym dictionary t_{text}:

$$\begin{aligned} &bd = t_{text} \cup Unk, \\ &t_{text} = \{a_1,...,a_n\} \cup \{a_{unk_1},...,a_{unk_m}\}. \end{aligned} \tag{6}$$

The values of the attributes of the consolidated data tuple are divided into the following:

1 Clear (known) - values of the primary key, foreign keys (may be missing). They are denoted by A.
2 Missing - physically missing information. They are denoted by $\perp$.
3 Undefined - a set of Unk attributes is entered for attribute subsets that indicate the degree of trust of the values of these attributes. By default, we assign the value of the Unk attribute to the highest degree of trust.

A Big data entity is a set of entity characteristic values described as:

$$bd = < A, A\_unk, Unk, \{dic\}, \{cg\} >, \tag{7}$$

where A is a subset of attribute values with clear values, $A = t_{rel} \cup t_{dw} \cup t_{text}$, A_unk is a subset of attribute values with fuzzy and non-deterministic values, $\boldsymbol{Unk}$ is a subset of attribute values with the degrees of trust of the attribute values values $\boldsymbol{A\_unk}$ and, $rel(A\_unk, U\_unk) = 1$, $\{dic\}$ is the set of data dictionary values, $\{cg\}$ is the set of data directory values.

Big data dataset bd is the set of data sources with schema *Bd* (1) and set of Big data entities ***bd*** (7).

The developed model enables to process the semistructured data. Having the Big data model we can run a next stage developing the method for recovering the missing data.

3.2. The Method of Missing Data Recovery

The proposed method of missing data recovery is based on functional dependencies and association rules and consists of two parts: (i) Probabilistic Production Dependencies mining based on Big data interface; (ii) Using of the Probabilistic Production Dependencies for missing data recovery.

3.2.1. Probabilistic Production Dependencies Mining

Analyzing large amounts of data requires identifying attribute groups that form functional dependencies (FD). However, in the real world, data sets are much more common, with important dependencies defined only on a subset of key attribute group values. Moreover, the dependencies can appear not only for tuples in relational data sources, but also between different parts of different tuples. We will call them Probabilistic Production Dependency (PPD).

Probabilistic Production Dependency is a production rule in the basic ratio selection that is valid for a significant number of entities in that selection. The significance threshold should be determined expertly, or based on calculations of the probability of erroneous selection of this dependence. The main difference between associative rules and PPD is that PPD will generated from existing FD in dataset.

$$\begin{aligned} &F_l :\ K = \{a_i\}, a_i \in A, D = \{a_j\}, \\ &a_j \in A, : P(k \in K \to d \in D) = p, \end{aligned} \tag{8}$$

where k and d are the tuples of groups of attributes K and D, respectively.

The main indicator of the reliability of such a dependency is the ratio of the number of objects that such a

PPD has to the number of objects in the selection:

$$P(F_I) = \frac{\left|\sigma_{k\in K \wedge d\in D}(R)\right|}{\left|\sigma_{k\in K}(R)\right|}. \quad (9)$$

The classification rule is called the probabilistic productive relationship between the subsets of the *X* and *Y* attributes in the consolidated Big data *Bd*, which occurs in training set *bd* with trust level *s*, where:

$$(X = x) \to (Y = y). \quad (10)$$

The classification rule is built on the training dataset with schema *Bd* with known class label values. This rule is built for schema *Bd* and therefore will not be affected by new entity coming to the Big data dataset (testing dataset independence).

A class label is called a linguistic variable or habitual characteristic of objects, which are values of a subset of attributes *Y* and denotes objects with common (like trust degree *s*) values of a subset of attributes of *X*. Attribute domains belonging to a subset of *Y*, $y \in dom(Y) = \pi_y(Bd)$ must contain a finite and known set of values.

Class labels are selected from a known set of values (within the study area are fixed), and the class of objects that have just been entered into the consolidated data repository is based on classification rules [22]. Tags will be added automatically as new data sources are also added to the catalog of Big data.

The calculation of implementation reliability for this dependence is based on the possibility of decomposition of such dependence into components of the PPD:

$$P(s\in S \to t\in T) = \sum_{t_i\in T} P(s\in S \to t = t_i) = \\ = \sum_{t_i\in T} \frac{\sum_j \left|s = s_j \wedge t = t_i\right|}{\sum_j \left|s = s_j\right|}. \quad (11)$$

As in the case of F-dependencies (functional dependencies), the set of classification rules that take place in a given relation can be represented by some subset of them, and then all classification rules of a given relation can be obtained by means of output rules. Since classification rules are an extension of F-dependencies, it is worth to consider the following axiom transformations of functional dependencies for classification rules:

Reflexivity. $P(s\in S \to s\in S) = 1$ for any source *rel*(*Bd*).

Proof:

$$P(s\in S \to s\in S) = \frac{\left|\sigma_{s\in S \wedge s\in S}\right|}{\left|\sigma_{s\in S}\right|} = \frac{\left|\sigma_{s\in S}\right|}{\left|\sigma_{s\in S}\right|} = 1. \quad (12)$$

Replenishment. If $P(s\in S \to t\in T) = p$, then $P(s\in S \wedge w\in D(W) \to t\in T) = p$.

Proof:

$$P(s\in S \wedge w\in D(W) \to t\in T) = \frac{\left|\sigma_{s\in S \wedge w\in D(W) \wedge t\in T}(R)\right|}{\left|\sigma_{s\in S \wedge w\in D(W)}(R)\right|} = \\ = \left|\forall x\in r : q = \pi_{W=w}(x) \in D(W) \Rightarrow w\in D(W)\right| = \\ = \frac{\left|\sigma_{s\in S \wedge t\in T}(R)\right|}{\left|\sigma_{s\in S}(R)\right|} = P(s\in S \to t\in T) = p. \quad (13)$$

Additivity. If $P(s\in S \to t\in T) = p$ and $P(s\in S \to w\in W) = 1$, then $P(s\in S \to t\in T \wedge w\in W) = p$.

Proof:

$$P(s\in S \to t\in T \wedge w\in W) = \frac{\left|\sigma_{s\in S \wedge t\in T \wedge w\in W}\right|}{\left|\sigma_{s\in S}\right|} = \\ = \left|s\in s \to w\in W\right| = \frac{\left|\sigma_{s\in S \wedge t\in T}\right|}{\left|\sigma_{s\in S}\right|} = P(s\in S \to t\in T) = p. \quad (14)$$

Metrics similar to associative rules are the important characteristics for PPD.

Trust Level is the ratio of the number of objects that have such a PPD to the number of objects in the selection:

$$Conf(S \to T) = P(S \to T) = \frac{\left|\sigma_{S\wedge T}(r)\right|}{\left|\sigma_S(r)\right|}. \quad (15)$$

Support Level is the characteristic of a selection predicate in a ratio that is calculated as the ratio of the number of objects that satisfy *P* predicate to the total number of objects in relation to:

$$Supp(P) = \frac{\left|\sigma_P(r)\right|}{|r|}. \quad (16)$$

When calculating the level of support for PPD, the conditional and the resulting dependency predicate are combined by a conjunction:

$$Supp(S \to T) = Supp(S \wedge T) = \frac{|\sigma_{S \wedge T}(r)|}{|r|}. \quad (17)$$

Using this concept, the *level of trust* can be calculated as:

$$Conf(S \to T) = \frac{Supp(S \to T)}{Supp(S)}. \quad (18)$$

The *level of improvement* is calculated as the ratio of the levels of trust and support of the PPD:

$$\mathrm{Imp}(S \to T) = \frac{Conf(S \to T)}{Supp(T)} = \frac{Supp(S \wedge T)}{Supp(S) \cdot Supp(T)}. \quad (19)$$

Total *mutual information* is generally defined as:

$$I_{X \leftrightarrow Y} = \sum_{i=1}^{n} \sum_{j=1}^{m} P_{ij} \log_2 \frac{P_{ij}}{p_i r_j}, \quad (20)$$

where $P_{ij} = P(X \sim x_i \wedge Y \sim y_j)$ is the probability that X is in condition x_i ,and Y is in condition y_j; $p_i = P(X \sim x_i)$ is the probability that X is in condition x_i; $r_j = P(Y \sim y_j)$ is the probability that Y is in condition y_j.

For associative rules, the *mutual information* will be defined as:

$$I_{X \leftrightarrow Y} = \sum_{i=1}^{n} \sum_{j=1}^{m} Supp(x_i \to y_j) \log_2 Imp(x_i \to y_j). \quad (21)$$

In the particular (binary) case, this characteristic can be calculated on the basis of three known characteristics: *Sup*, *Conf*, *Imp* [24].

The uncertainties and missing data that occur among the values of the attribute *Y* of the relation *r* are classified using PPD. Taking the all above, the following algorithm for PPD building is designed.

```
Algorithm 1. PPD mining algorithm
Input: Big data dataset bd, Card(Bd)=n
Output: PPDlist
1:PPDlist={}
2:X={}
3:for (i=1;i<n;i++)
4:      $X = X \cup A_i$
5:      Group entities with the same values for the set of attributes X;
6:      Search for entities that have the same values for the set of synonyms of attributes X;
7:      Y={}
8:      for (j=i+1;j<=n;j++)
9:      $Y = Y \cup A_i$
10:        Calculating Supp and Conf in the source of the entities obtained in steps 5) and 6);
11:        Calculating the Imp of the tuple sources obtained in steps 5) and 6);
12:        Identifying the entities with the highest levels of confidence and add X→Y to PPDlist.
```

3.2.2. Using of the Probabilistic Production Dependency for Missing Data Recovery

To classify objects (or fill in missing data), we need to build classification rules. Generally, Big data can store information about several types of classes, and there is a subset of functions per each class type. The same function can be employed to define several types of classes.

Classification rules are called PPDs that are performed for a certain subset of the tuples for consolidated Big data *bd*. The following algorithm for data imputation is designed.

```
Algorithm 2. Recovery algorithm
Input: Big data dataset bd with schema Bd and missed data ⊥, PPDList
Output: Completeness level of bd
1:Completeness=0
2:If $(bd) = \{bd_1(X_1)\downarrow, ..., bd_1(X_n)\downarrow\}$ and $\{bd_2(X_1)\downarrow, ..., bd_2(X_n)\downarrow\}$ and $\{bd_1(X_1)\downarrow, ..., bd_1(X_n)\downarrow = bd_2(X_1)\downarrow, bd_2(X_n)\downarrow\}$ and $bd_1(Y)\downarrow$ and $bd_2(Y) = \perp\}$ and $\sigma_{X_1}(Dic) = \varnothing$ and $\{X_1 \to Y\ in\ \ PPDlist\}$
3:then
        change ⊥ to $bd_1(Y)$
        $bd_1(P) = bd_1(P) / \left(\sum_i \frac{m_{1i}}{n}\right)$
        Completeness++
4:If $\{bd_1(X_1)\downarrow, ..., bd_1(X_n)\downarrow\}$ and {m from n values of attributes are ↓ in $bd_2$, n - m values of attributes are ⊥, $m \le n$} and $\{P \ge 1 - m/n\}$
```

```
and {using defined values
$bd_1(X^m)\downarrow$, ..., $bd_2(X^m)\downarrow\}$ } and
$bd_1(Y)\downarrow$, ..., $bd_2(Y) = 1\}$
and {$X_2 \to Y$ in PPDlist}
5:then
      change ⊥ to $bd_2(Y)$
      $bd_2(P) = bd_2(P) / \left( \sum_i \frac{m_{2i}}{n} \right)$
6: If { $m_i$ from $n$ values of attributes are
$\downarrow$ in $bd_i$, $m_i \le n$ }
and { $m_j$ from $n$ values of attributes are $\downarrow$
in $bd_j$, $m_j \le n$ }
and {for exact values $bd_i(X^m)\downarrow = bd_2(X^m)\downarrow\}$
and {for exact values $bd_j(X^m)\downarrow = bd_2(X^m)\downarrow$}
and $\left\{ {}^{m_i}/_{n} \le {}^{m_j}/_{n} \right\}$ and $\left\{ P \ge 1 - {}^{m_i}/_{n} \right\}$,
and $\{bd_i(Y)\downarrow\}$ and $\{bd_j(Y)\downarrow\}$ and $\{bd_2(Y) = \perp\}$,
and $\left\{ X_2, X_j \to Y \ in \ PPDlist \right\}$
7:then
      change ⊥ to $bd_j(Y)$
      Completeness++
```

The sequence of often occurred events is important for the analysis too. If patterns are identified in such sequences, it is possible to predict with some degree of probability the occurrence of events in the future. This sequence also enables eliminating the missing data. Then the sequence of objects can be described in the following form:

$$S = \left\{ \ldots, i_p, \ldots, i_q \right\}, \text{where } p < q. \tag{22}$$

For example, such a sequence of objects may be the submission date of the medical tests. The following sequence:

$$S = \left\{ \begin{matrix} (hemoglobin\ level,\ 01.10.2019), \\ (pressure,\ 25.09.2019),\ (pH,\ 28.09.2019) \end{matrix} \right\} \tag{23}$$

can be interpreted as a sequence of assays by one person at different times (first, the venous pressure was measured, then the pH level was measured, and finally, the hemoglobin level was defined).

There are *two* types of sequences: with cycles and without cycles. For the *first* type, we can enter the sequence of the same object in different positions:

$$S = \left\{ 1 \ldots, i_p, \ldots, j_p, \ldots \right\}, \text{where } p < \text{q}, i_p = j_p. \tag{24}$$

For the *second* type, we can enter the sequence of the different object in same positions:

$$S = \left\{ 1 \ldots, i_p, \ldots, j_p, \ldots \right\}, \text{where } i_p < j_p. \tag{25}$$

Transaction T contains the sequence S if $S \subseteq T$ and the objects included in S belong to the set T with the order relation being preserved. It is assumed that other objects between the objects in the sequence S may be allocated in the set T there.

A support of the sequence S is the ratio of the number of transactions contenting S to the total number of transactions. A sequence is commonly used if its support exceeds the minimum user defined support:

$$Supp(S) > minSupport. \tag{26}$$

To recover the missing data, we must find these sequences and use them to empty data filling.

Let us assume that a *bd* entities characterize an object, its condition in time or its property. Then modeling the object in the Big dataset *Bd* is run by establishing a relation between the individual tuples. In addition, since each object is a system of some complexity, the properties of the object components can determine the properties of the object itself, or, conversely, transfer the properties of the object in its components.

Having the operators for transition from sub characteristic to meta characteristic we may build the sequence of rules as well as use this sequence for recovery of missing data. It enables creating the statistical tree and using for PPD generation.

Now we determine the *parent moving* operator

$$Up_{c_{X=x_1,\sigma_X(Dic)}(bd)} = \sigma_{X=x_1}(\sigma_{X=Y,\sigma_X(Dic)}(bd)), \tag{27}$$

where x_1 is the value of primary key (child); x_2 is the foreign key (parent), and

the *child moving* operator

$$Down\_c_{X=x_2,\sigma_X(Dic)}(bd) = \sigma_{Y=x_2,\sigma_X(Dic)}(bd). \tag{28}$$

Based on the above described, we may build the operator for recovering the missing data in the sequence of the events (or entities):

$$\sigma_{X=x_1,Val=v,\sigma_X(Dic)}(bd) = = Heir\_c\begin{pmatrix} \sigma^{cons}_{X=x_1,Val=v,\sigma_X(Dic)}(bd), \\ Down\_c_{X=x_1,Val=Null,\sigma_X(Dic)}(bd) \\ Bd.Unk_X = \\ = Recovery\left(Bd.Unk_X, P^X\left(\sigma_X\left(Dic\right)\right)\right) \end{pmatrix}. \tag{29}$$

3.3. Estimation of Algorithm Complexity for Missing Data Recovery

The proposed algorithm of data analysis includes the two consecutive steps. Accordingly, the asymptotic complexity of the whole algorithm will be the sum of complexity components for individual steps:

$$t = O\left(t_{stat} + t_{ma}\right), \tag{30}$$

where t is time complexity of algorithm for missing data recovery.

We will evaluate the asymptotic complexity separately per each stage (Stage I and Stage II).

Stage I

At this stage, the input relation with the data is passing through each tuple, so the time of this stage is directly proportional to the relation size n.

During each tuple transmission, a hierarchy of attributes is kept with corresponding values of the tuples number in the added tree branches. Since each vertex of the tree corresponding to the projection on i attribute may contain m children, traversing the whole tree (29) in the worst case will take

$$O\left(m\cdot\left|D\left(A_1\right)\right|\cdot(m-1)\cdot\left|D\left(A_2\right)\right|\cdot(m-2)\cdot\left|D\left(A_3\right)\right|\cdot\ldots\cdot1\cdot\left|D\left(A_m\right)\right|\right) = = O\left(m!\cdot\prod_{i=1}^{m}\left|D\left(A_i\right)\right|\right), \tag{31}$$

where m is amount of attributes, $\left|D\left(A_2\right)\right|$ is amount of unique values of i attribute. It means that the worst case of computational complexity coincides with the FP-tree method for the construction of associative rules. The proposed algorithm can only be practical for data with a small number of attributes and valid values per each attribute.

The developed method assumes the existence of a *minSupport* constraint. In case of attributes independence, it enables reducing the height of the statistical tree from $O(m)$ to

$$O\left(\log_{\sqrt[m]{\prod_{i=1}^{m}\left|D\left(A_i\right)\right|}}\left(\frac{n}{minSupport}\right)\right), \tag{32}$$

where n is the number of tuples for relation.

In the partial case, when all attributes have one domain and evenly distributed values, the asymptotic of the height for the statistics tree

$$O\left(\log_{D\left(A_m\right)}\left(\frac{n}{minSupport}\right)\right). \tag{33}$$

Each node of tree has $O\left(m\cdot\left|D\left(A\right)\right|\right)$ direct children. Accordingly, the number of tree nodes can be estimated by expression:

$$V_{stat} = O\left(\left(m\cdot\left|D\left(A\right)\right|\right)^{\log_{avgD}\left(\frac{n}{minSupport}\right)}\right) = \\ = O\left(m^{\log_{avgD}\left(\frac{n}{minSupport}\right)}\cdot\frac{n}{minSupport}\right) = \\ = O\left(\frac{n}{minSupport}^{\log_{avgD}(m)}\cdot\frac{n}{minSupport}\right) = \\ = O\left(\left(\frac{n}{minSupport}\right)^{1+\log_{avgD}(m)}\right). \tag{34}$$

For most practical data analysis tasks, it is advisable to set the *minSupport* value commensurable with n. For example, if $minSupport = \sqrt{n}$, then

$$V_{stat} = O\left(n^{\frac{1+\log_{avgD}(m)}{2}}\right). \tag{35}$$

Thus, the size of the statistics tree is highly dependent on the structure of the stored data domains. If their domains have a high power and evenly distributed data, then the number of nodes can be estimated by $O\left(\sqrt{n}\right)$. If the power of domains is close to the number of analyzed attributes, then the statistics tree grows to $O(n)$ nodes.

A further increase of attributes number and the insufficient variation of data will lead to the dramatic growth of nodes number in statistics tree. If there is a need to analyze the data of such a structure, it is recommended that *minSupport* be set to a fixed fraction of attributes number for the relation: $minSupport = q \cdot n,\ q \in (0;1)$. This will stabilize the number of nodes in statistics tree and store it in memory during the process of data analyzing:

$$V_{stat} = O\left(q^{-\frac{1}{2}\cdot\left(1+\log_{avgD}(m)\right)}\right). \qquad (36)$$

Let us evaluate the computational complexity of building such a statistics tree. For each tuple, we need to find the nodes of a tree that match its attribute values and increment the number of data records, which meet those criteria. The first level of the tree preserves the conditions $A_1 = v_{1j}, \ldots, A_2 = v_{2j}, \ldots, A_m = v_{mj}$, that is, projection onto specific values of one attribute. Accordingly, the tuple will correspond to m nodes of the first level (the tuple has its unique meaning per each attribute). The tops of the second level correspond to the projections:

$$\begin{aligned} &A_1 = v_{1j} \wedge A_2 = v_{2j},\ A_1 = v_{1j} \wedge A_3 = v_{3j}, \ldots, \\ &A_1 = v_{1j} \wedge A_m = v_{mj}, A_2 = v_{2j} \wedge A_3 = v_{3j}, \\ &A_2 = v_{2j} \wedge A_4 = v_{4j}, A_2 = v_{2j} \wedge A_m = v_{mj}, \ldots, \\ &A_{m-1} = v_{(m-1)j} \wedge A_m = v_{mj} \end{aligned} \qquad (37)$$

that is, all possible projections to the specific values of the attribute pair. At this level, the tuple can fit maximum m^2 conditions (some top-tier peaks may not have enough support for sub-conditions to be smaller than *minSupport*). Similarly, at the third level, we get a limited number of nodes matched with the tuple: $O\left(m^3\right)$, on the l-th level – $O\left(m^l\right)$.

Hence, the total computational complexity of adding one tuple to the statistics tree:

$$t_c = O\left(1 + m + m^2 + \ldots + m^{H_{stat}}\right) = O\left(m^{H_{stat}}\right). \qquad (38)$$

Accordingly, we can estimate the processing of data array from n tuples:

$$\begin{aligned} t_{stat} &= O\left(n \cdot m^{H_{stat}}\right) = O\left(n \cdot m^{\log_{avgD}\left(\frac{n}{minSupport}\right)}\right) = \\ &= O\left(n \cdot \left(\frac{n}{minSupport}\right)^{\log_{avgD}(m)}\right) = \\ &= O\left(minSupport \cdot \left(\frac{n}{minSupport}\right)^{1+\log_{avgD}(m)}\right). \end{aligned} \qquad (39)$$

As it can be seen from above, the computational complexity of collecting statistics can depend linearly on the size of the input ratio when limiting factors are selected successfully. Getting the analyzed dependence on attributes in more detail, we can conclude that the height of the statistics tree exceeds the 3 for a small number of branches only. Therefore, we can use a rough estimate $H_{start} \approx 2$ and the dependence of computational complexity on the number of attributes is quadratic one.

Let us consider the complexity of the algorithm for collecting the memory usage statistics. Since the statistics tree is the persistent data structure only, the number of tree nodes estimate a volume of the required memory

$$M_{stat} = O\left(\left(\frac{n}{minSupport}\right)^{1+\log_{avgD}(m)}\right). \qquad (40)$$

Stage II

If we use the Functional Dependencies as Elemental Dependencies for PPD Generation Enables Complexity, then we get:

$$t_{ma} = O\left(\frac{Z_{aggr}^2 \cdot \log\left(sz_{aggr}\right)}{m \cdot D(A)}\right). \qquad (41)$$

The optimization is made determining many dependencies with the same result part or the same conditional part per each dependency, through the hash table. The combination does not occur with all other elementary dependencies but with the corresponding merge condition only. The structure of the data and the specific sets of dependencies will play a significant role in this process.

The number of stored dependencies and the number of defined attribute values determine the volume of the required memory

$$M_{ma} = O\left(Z_{ma} \cdot sz_{ma}\right), \tag{42}$$

where sz_{ma} is the average power of attribute values set that determine the predictors of the condition and the result of PPD presentation.

Thus, the proposed algorithm enables performing the effective data analysis for PPD presence, and its total complexity

$$t = O\left(\begin{array}{l} minSupport \cdot \left(\dfrac{n}{minSupport}\right)^{1+\log_{avgD}(m)} + \\ + \dfrac{Z_{el}^2}{m \cdot D(A)} + \dfrac{Z_{aggr}^2 \cdot \log\left(sz_{aggr}\right)}{m \cdot D(A)} \end{array}\right) = \\ = O\left(\begin{array}{l} minSupport \cdot \left(\dfrac{n}{minSupport}\right)^{1+\log_{avgD}(m)} + \\ + \dfrac{Z_{aggr}^2 \cdot \log\left(sz_{aggr}\right)}{m \cdot D(A)} \end{array}\right). \tag{43}$$

4. Experimental Results and Discussion

To run the experiment, the dataset from the Big Cities Health Inventory Data Platform ("BCHC Data Platform") is used. The platform contains over 18,000 data points across more than 50 health, socio-economic, urban (information from smart sensors) and demographic indicators across 11 categories in the United States. This is an example of semictructured data with hidden dependencies inside entities and between entities.

The missing data are modeled as random deleting of exact data.

The proposed and existing methods are tested on the same hardware: Intel Core 5 Quad E6600 2.4 GHz, 16 GB RAM, HDD WD 2 TB 7200 RPM. The criteria of PPD creation are $Conf(F_1)>0.7$ and $minSupport=100$. RStudio is used for data modelling and analysis.

The proposed method was compared with the existing ones: AR, RF, SVM, MLP, EM and KNN (Figure 1), where the recovery error is presented by the normalized root-mean-square error (NRMSE).

As it can be seen from Figure 1, the proposed PPD method is the most precision among all methods, and the EM classifier gives the closest results.

As it can be seen from the analysis results (Figure 2), in the range of about 1%-40% missed data the proposed PPD method looks better than most of existing methods. For example, the proposed PPD method performs 12% better than the EM and RF methods for 30% missing data. For more percentage of missed data (the range about 40%-50% missed data) the EM method looks the best and the PPD has comparable results with the SVM method.

Results of time complexity analysis for proposed and existing methods are given in Table 1.

Figure 1
NRMSE recovery error

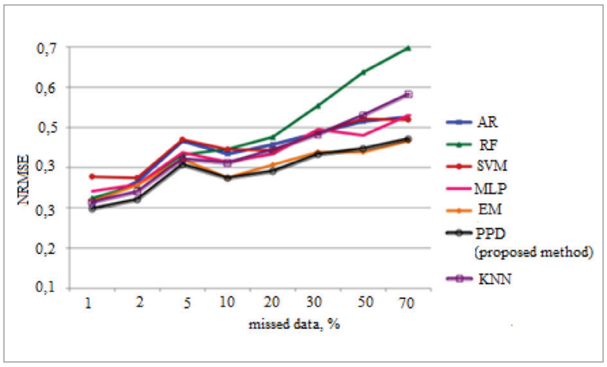
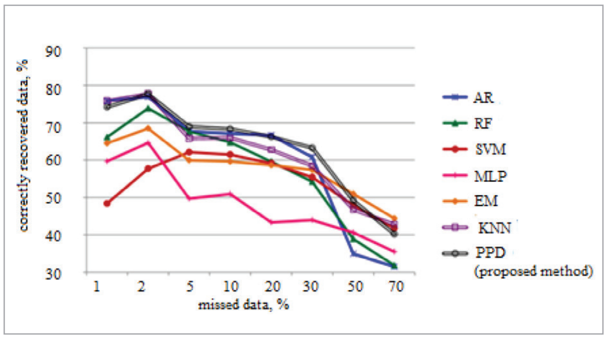


Figure 2
The analysis of the percentage of correctly recovered data



The time complexity is calculated for model generation stage along with missing data imputation stage. As it can be seen from Table 1, the time complexity of PPD method is the smallest and the EM algorithm has shown the closest result to PPD.

Table 1
Time of analysis (*min*), depending on the amount of analyzed data

Amount of records	SVM	AR	EM	PPD (proposed method)
2 000	31	190	9	8
4 000	49	362	23	19
6 000	95	437	37	31
8 000	144	639	49	42
10 000	210	827	62	51
18 000	286	1019	77	65

As the number of attributes grows, the number of found dependencies increases much faster than when the volume of the analyzed data grows, and this conclusion is not restricted by described above.

Let us analyze the algorithm complexity in parallel mode considering $\frac{n}{k}$ tuples per each from k servers (processors). In such case, the algorithm complexity should be evaluated for the number of features *m*=-Card(*Bd*) in dataset *Bd,* and its time complexity:

$$t_k = O\left(\begin{array}{l} minSupport \cdot \left(\dfrac{n}{k \cdot minSupport} \right)^{1+\log_{avgD}(m)} + \\ +\log_2(k) \cdot \left(\dfrac{n}{k \cdot minSupport} \right)^{1+\log_{avgD}(m)} \end{array} \right) + $$
$$+O\left(\frac{Z_{aggr}^2 \cdot \log\left(sz_{aggr}\right)}{k \cdot m \cdot D(A)} \right) = .$$
$$= O\left(\begin{array}{l} \left(minSupport + \log_2(k)\right) \cdot \left(\dfrac{n}{k \cdot minSupport} \right)^{1+\log_{avgD}(m)} + \\ + \dfrac{Z_{aggr}^2 \cdot \log\left(sz_{aggr}\right)}{k \cdot m \cdot D(A)} \end{array} \right) \quad (44)$$

If $k = O(n)$, then

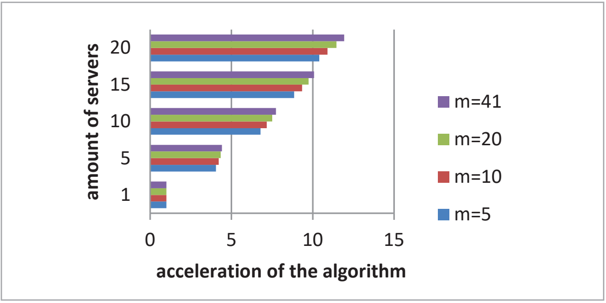
$$t_n = O\left(\begin{array}{l} \left(minSupport + \log_2(n)\right) \cdot minSupport^{-1-\log_{avgD}(m)} + \\ + \dfrac{Z_{aggr}^2 \cdot \log\left(sz_{aggr}\right)}{n \cdot m \cdot D(A)} \end{array} \right). \quad (45)$$

The amount of generated PPD Z_{aggr} is equal to thousands and the amount of tuples is equal to billions. Therefore, in the case of parallel computing on a large number of processors, we can neglect by the second member of the function t_n. For the same reason, $log_2(n) = O(minSupport)$. Thus, an asymptotic estimate of the algorithm execution time (on a system of *k* processors)

$$t_n = O\left(minSupport^{-\log_{avgD}(m)} \right). \quad (46)$$

The function (46) has a sub-polynomial character. The developed algorithm enables to assert that the issue of detecting PPD in Big data belongs to the class of P-problems with a reason to employ the high performance computing. Authors explored the ways of algorithm acceleration using the different number of servers (Figure 3).

Figure 3
The acceleration graph of the developed algorithm



As it can be seen from Figure 3, the acceleration for m=41 attributes is larger in 12.5 times for 20 servers (processors) compared to the non-parallel mode. The acceleration of the algorithm depends also on the number of attributes. For example, the acceleration for 20 servers (processors) is equal to 12.5 for 41 attributes and 10.5 for 5 attributes (see Figure 3). Thus, the acceleration of PPD algorithm is higher in about 1.2 times for large datasets (41 attributes) than for small dataset (5 attributes).

5. Conclusion

The authors proposed a three-stage approach for missing data imputation, which includes: (i) designing the Big data model in the task of missing data recovery, which enables to process the semistructured data; (ii) developing the method of missing data recovery based on functional dependencies and association rules, and (iii) estimating the algorithm complexity for missing data recovery.

The proposed method of missing data recovery creates additional data values using a based domain and functional dependencies and adds these values in available training data. The correctness of the imputed values is verified on the classifier built on the original dataset. The proposed PPD method performs 12% better than the EM and RF methods for 30% of missing data. For more percentage of missing data (the range about 40%-50% missing data) the EM method looks the best, and the PPD has comparable results with SVM method. Comparison of methods for analyzing and modeling the statistical processes by qualitative criteria confirmed the developed method has following advantages: possessing the characteristics of resistance to errors in the data, enabling the parallel execution in distributed databases, automating and performing the analysis of different types of data. The acceleration for m=41 attributes is larger in 12.5 times for 20 servers (processors) comparing with the non-parallel mode.

Further research will be dedicated to recovering the missed values in streaming data. The sources of such data are various types of sensors, actuators etc. Therefore, data recovery without buffering can increase the speed of data processing.

Acknowledgments

This work is funded by the National Natural Science Foundation of China under Grant No. 61772180, Technological innovation project of Hubei Province 2019(2019 AAA047), Green Industry Science and Technology Leadership Program of Hubei University of technology (No.CPYF2018005).

References

1. Ahmat Zainuri, N., Jemain, A. A., Muda, N. A Comparison of Various Imputation Methods for Missing Values in Air Quality Data. JSM, 2015, 44, 449-456. https://doi.org/10.17576/jsm-2015-4403-17
2. Alhroob, A., Alzyadat, W., Almukahel, I., Altarawneh, H. Missing Data Prediction Using Correlation Genetic Algorithm and SVM Approach. Population, 2020, 11(2). https://doi.org/10.14569/IJACSA.2020.0110288
3. Aydilek, I. B., Arslan, A. A Hybrid Method for Imputation of Missing Values Using Optimized Fuzzy C-Means with Support Vector Regression and a Genetic Algorithm. Information Sciences, 2013, 233, 25-35. https://doi.org/10.1016/j.ins.2013.01.021
4. Christopher, S. Z., Siswantining, T., Sarwinda, D., Bustaman, A. Missing Value Analysis of Numerical Data using Fractional Hot Deck Imputation. Proceedings of 3rd International Conference on Informatics and Computational Sciences (ICICoS), 2019, 1-6. https://doi.org/10.1109/ICICoS48119.2019.8982412
5. Hayati Rezvan, P., Lee, K. J., Simpson, J. A. The Rise of Multiple Imputation: A Review of the Reporting and Implementation of the Method in Medical Research. BMC Med Res Methodol, 2015, 15, 30. https://doi.org/10.1186/s12874-015-0022-1
6. Izonin, I., Tkachenko, R., Kryvinska, N., Zub, K., Mishchuk, O., Lisovych, T. Recovery of Incomplete IoT Sensed Data using High-Performance Extended-Input Neural-Like Structure. Procedia Computer Science, The 10th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN-2019), 2019, 160, 521-526. https://doi.org/10.1016/j.procs.2019.11.054
7. Jung, S., Moon, J., Park, S., Rho, S., Baik, S. W., Hwang, E. Bagging Ensemble of Multilayer Perceptrons for Missing Electricity Consumption Data Imputation. Sensors, 2020, 20(6), 1772. https://doi.org/10.3390/s20061772
8. Jurgelevičius, A., Sakalauskas, L. Big Data Mining Using Public Distributed Computing. Information Technology and Control, 2018, 47(2), 236-248. https://doi.org/10.5755/j01.itc.47.2.19738
9. Karahalios, A., Baglietto, L., Carlin, J. B., English, D. R., Simpson, J. A. A Review of the Reporting and Handling of Missing Data in Cohort Studies with Repeated Assessment of Exposure measures. BMC Med Res Methodol, 2012, 12, 96. https://doi.org/10.1186/1471-2288-12-96
10. Kowarik, A., Templ, M. Imputation with the R Package. VIM, 2016. https://doi.org/10.18637/jss.v074.i07
11. Leke, C. A., Marwala, T. Introduction to Missing Data Estimation. Deep Learning and Missing Data in Engineering Systems 2019, 1-20. https://doi.org/10.1117/12.2053057

12. Lim, S. Y., Mohamad, M. S., Chai, L. E., Deris, S., Chan, W. H., Omatu, S., Ibrahim, Z. Investigation of the Effects of Imputation Methods for Gene Regulatory Networks Modelling Using Dynamic Bayesian Networks. Distributed Computing and Artificial Intelligence, 13th International Conference, 2016, 413-421. https://doi.org/10.1007/978-3-319-40162-1_45

13. Lin, T. H. A Comparison of Multiple Imputation with EM Algorithm and MCMC Method for Quality of Life Missing Data. Quality and Quantity, 2010, 44, 277-287. https://doi.org/10.1007/s11135-008-9196-5

14. McPherson, S., Barbosa-Leiker, C., Mamey, M. R., McDonell, M., Enders, C. K., Roll, J. A "Missing not at Random" (MNAR) and "Missing at Random" (MAR) Growth Model Comparison with a Buprenorphine/naloxone Clinical Trial. Addiction, 2015, 110, 51-58. https://doi.org/10.1111/add.12714

15. Davar, G., Abdi, M. J. Automatic Detection of Erythemato-Squamous Diseases Using PSO-SVM Based on Association Rules. Engineering Applications of Artificial Intelligence, 26(1), 2013, 603-608. https://doi.org/10.1016/j.engappai.2012.01.017

16. Myers, T. A. Goodbye, Listwise Deletion: Presenting Hot Deck Imputation as an Easy and Effective Tool for Handling Missing Data. Communication Methods and Measures, 2011, 5, 297-310. https://doi.org/10.1080/19312458.2011.624490

17. Nelwamondo, F. V., Mohamed, S., Marwala, T. Missing Data: A Comparison of Neural Network and Expectation Maximisation Techniques, 2007, arXiv:0704.3474 [stat].

18. Pedersen, A. B., Mikkelsen, E. M., Cronin-Fenton, D., Kristensen, N. R., Pham, T. M., Pedersen, L., Petersen, I. Missing Data and Multiple Imputation in Clinical Epidemiological Research. Clinical Epidemiology, 2017, 9, 157-166. https://doi.org/10.2147/CLEP.S129785

19. Rahman, Md. G., Islam, M. Z. Missing Value Imputation Using Decision Trees and Decision Forests by Splitting and Merging Records: Two Novel Techniques. Knowledge-Based Systems, 2013, 53, 51-65. https://doi.org/10.1016/j.knosys.2013.08.023

20. Raković, L., Sakal, M., Matković, P., Marić, M. Shadow IT-Systematic Literature Review. Information Technology and Control, 2020, 49(1), 144-160. https://doi.org/10.5755/j01.itc.49.1.23801

21. Sachenko A., Kochan V., Turchenko V. Instrumentation for Gathering Data. IEEE Instrumentation and Measurement Magazine, 2003, 6 (3), 34-40. https://doi.org/10.1109/MIM.2003.1238339

22. Shakhovska, N., Kaminskyy, R., Zasoba, E., Tsiutsiura, M. Association Rules Mining in Big Data. International Journal of Computing, 2018, 17(1), 25-32. http://www.computingonline.net/computing/article/view/946

23. Shakhovska, N., Strubytskyi, R. Model of Data Warehouse with Uncertain Consolidated Data. Applied Mathematics & Information Sciences, 2015, 9(4), 1753. http://dx.doi.org/10.12785/amis/090412

24. Shakhovska, N., Vovk, O., Kryvenchuk, Y. Uncertainty Reduction in Big Data Catalogue for Information Product Quality Evaluation. Eastern-European Journal of Enterprise Technologies, 2018, 1, 12-20. https://doi.org/10.15587/1729-4061.2018.123064

25. Shi, D., Lee, T., Fairchild, A. J., Maydeu-Olivares, A. Fitting Ordinal Factor Analysis Models with Missing Data: A Comparison Between Pairwise Deletion and Multiple Imputation. Educational and Psychological Measurement, 2020, 80, 41-66. https://doi.org/10.1177/0013164419845039

26. Siroky, D. S. Navigating Random Forests and Related Advances in Algorithmic Modeling. Statistics Surveys, 2009, 3, 147-163. https://doi.org/10.1214/07-SS033

27. Stamatescu G., Făgărăşan I., Sachenko A. Sensing and Data-Driven Control for Smart Building and Smart City Systems. Journal of Sensors, 2019, Article ID 4528034, 3 pages, https://doi.org/10.1155/2019/4528034