STATISTICS, OPTIMIZATION AND INFORMATION COMPUTING
Stat., Optim. Inf. Comput., Vol. **9**, September 2021, pp 665–680.
Published online in International Academic Press (www.IAPress.org)

Automated Noise Detection in a Database Based on a Combined Method

Mahdieh Ataeyan , Negin Daneshpour *

Department of computer engineering, Shahid Rajaee Teacher Training University, Tehran, Iran

Abstract Data quality has diverse dimensions, from which accuracy is the most important one. Data cleaning is one of the preprocessing steps in data mining which consists of detecting errors and repairing them. Noise is a common type of error, that occur in database. This paper proposes an automated method based on the $k-$means clustering for noise detection. At first, each attribute (A_j) is temporarily removed from data and the $k-$means clustering is applied to other attributes. Thereafter, the $k-$nearest neighbors is used in each cluster. After that a value is predicted for A_j in each record by the nearest neighbors. The proposed method detects noisy attributes using predicted values. Our method is able to identify several noises in a record. In addition, this method can detect noise in fields with different data types, too. Experiments show that this method can averagely detect 92% of the noises existing in the data. The proposed method is compared with a noise detection method using association rules. The results indicate that the proposed method have improved noise detection averagely by 13%.

Keywords Data Cleaning, Automated Noise Detection, Clustering, K-means, Data Quality.

DOI:10.19139/soic-2310-5070-879

1. Introduction

The progressive increase in data has caused organizations to face a large amount of data as well as heterogeneous and distributed sources. These data are used in decision-making and knowledge acquiring. The potential business value of these decisions depends on the quality of data used to make them [31]. When data are transferred from different sources and systems to another system, errors or problems such as heterogeneous format or domain may arise. Making decisions based on the low quality data not only causes damages to the structure of the organizations but also imposes high costs to them. According to the studies, more than 30% of real world data lacks in quality leading to damages of three trillion dollars per year to the US government [26]. That is why the data quality is highly important in data sources. Although preparing high quality data is time and cost consuming [8], it is significantly better than making mistake because of the low quality data.

Data quality has various dimensions, from which accuracy is of higher importance. Problems such as noise, incompleteness, inconsistency, and missing values cause this dimension to be violated. Data correction is a process used to detect noisy, incomplete, and inconsistent data and improves data quality by correcting detected errors [15]. Data correction procedure may be boring and time-consuming, however, it cannot be overlooked [30]. Considering the high volume of data, interactive correction approaches are inapplicable and thus, automated approaches are required.

Data mining is a key technique for data correction [15]. Various approaches have been provided up to now for data correction using ontologies [5], classifiers such as decision trees [17, 16] and neural networks [4], rules [26, 13, 19], ensemble learning [4], Markov logic networks [10], functional dependencies [12] , and conditional

*Correspondence to: Negin Daneshpour (Email: ndaneshpour@sru.ac.ir). Department of computer engineering, Shahid Rajaee Teacher Training University, Tehran, Iran.

ISSN 2310-5070 (online) ISSN 2311-004X (print)

functional dependency [18]. Some of these approaches have their own disadvantages including user interaction, failure to detect errors, and inability to separate detection phase from correction one. Separating detection and correction phases allows a user to view detected errors and apply another method to correct some of them. The repairing data process consists of validation the format and domain of attributes, discovering inconsistencies using the relationship between attributes, and imputation missing values.

Considering that user interaction is necessary for some approaches [5] and user interaction is impossible in a large amount of data, this paper proposes an automated approach for noise detection. Some approaches are only able to detect errors in a special type of data [17, 16]. By contrast, the approach proposed in this paper is able to detect the noise in all data types (numeric, nominal, and ordinal). In addition, error detection phase and error correction phase cannot be distinguished from each other in some previous approaches and the user cannot use another approach to correct data [4]. In some approaches, it is supposed that errors are detected and therefore, the method just proposed a cleaning algorithm [32]. In the proposed approach, noises are only detected and any approach can be used to correct them. The aim of our approach is detection noisy data automatically which inconsistent with other data.

Technical contribution:The main contributions of this paper are:

(1). Proposing a new combination of clustering and classification algorithms to automatically detect noisy fields. At first, each attribute is removed from data set. Then, the $k-$means clustering is applied on the remaining attributes. After clustering, for each record in each cluster, k nearest records are chosen and predict a value for the removed attribute. Noisy attributes are detected based on predicted values. The procedure is applied for all attributes. The proposed method is able to detect more than one noisy field in each record, because each attribute is considered separately.

2). In this approach, two strategies are used for attributes with different data types (most repeated for ordinal and nominal attributes, and averaging for numeric attributes). Therefor, the proposed approach can be used to detect noisy attributes with different data types.

Some approaches are able to detect only one noise in a record, while the proposed approach is able to detect several noises in a record. This approach is based on the $k-$means clustering and the $k-$nearest neighbors classification. At first, attribute A_j is temporarily taken away from data and the $k-$means clustering is applied on remaining data. Then, the $k-$nearest neighbors is used in each cluster and a value is predicated for A_j in each record. Noisy records are detected using predicted values. This process is iterated for each attribute in data.

Results of experiments on five data sets with different attributes, taken from the UCI Machine Learning Database, indicate that this approach can averagely detects 92% of the noises in experimented data sets. In addition, the proposed approach has been compared with a similar approach which uses association rules [15] for noise detection. The aim of selecting this method for comparisons is its similarities with the proposed approach. It has only noise detection phase and detects noisy data set automatically with different data types. Moreover, the rule based approach detects noisy data set with different data types like our approach. The results show that the proposed approach can detect noises averagely 13% more than the compared method [15].

The next sections of this paper are as follows: in the section 2, the related works on error detection approaches are described and the disadvantages of each method are expressed. The section 3 explains the proposed approach. In the section 4, the results of experiments on five data sets are explained which have different data types. In addition, the proposed approach is compared with the rule-based error detection method [15] in the section 4. The conclusion of the paper is provided in the last section.

2. Related work

The data of the real world have their own problems such as inconsistency, noise, and missing values. Such an error may occur on different data types. To prevent the impact of these erroneous data on the decisions, the errors have to be corrected. The procedure of data correction consists of two phases: detection of the erroneous attributes and replacing it by correct values. Error detection includes detection of noisy data, inconsistencies among values of attributes and records, and missing values. After the detection phase, users can either delete or clean erroneous fields using correction techniques. Many of approaches provided in this field use data mining techniques for data

correction. Some of these approaches can detect and correct only a special type of data [17, 16]. By contrast, some other ones can detect and correct all types of data. In this section, some of these approaches and their disadvantages are explained.

The method proposed in [5] uses an ontology for detecting valid and invalid values in records. Invalid values are detected and provided a user together with all valid values existing in the ontology. After that, the system asked the user to select a value for erroneous field among valid values. The selected values by the user are saved together with invalid values. When the number of corrected erroneous fields by the user reaches a predetermined threshold level, rules are extracted from them. Thereafter, the extracted rules are used to correct fields which are detected by the ontology as erroneous fields. As the method need to interact with the user in the first phase, it is not appropriate for a large amount of data.

In [22] an approach has been proposed to impute missing data using grey based fuzzy c-means, mutual information based feature selection, and regression model. At first, this algorithm finds the priority of each missing attribute. Then, data are clustered using Fuzzy c-Means. Next, the algorithm chooses clusters which satisfy a minimum condition. After that, mutual information is used to select highly dependent features of instances within each cluster. To provide estimations for a missing value, regression models will be applied to the selected features. Finally, the missing value is imputed through a weighted average of estimated values obtained from the previous step.

CMIM [†] method estimates missing value using correlation maximization [23]. At first, a base set is selected from complete instances. This method uses ten algorithms to maximize correlation. Then, each maximization algorithm attempts to find subsets of data with strong correlations with respect to the missing attribute of a missing instance. Finally, a method imputes the missing value by applying regression model to the highly correlated subset.

DMI method has been developed based on EM [‡] algorithm and C4.5 decision tree [17, 16]. This method is used to impute missing values. For this purpose, records with missing values are detected and a decision tree is built for each attribute using complete records. The decision tree classifies records into several leaves in such a way that records existing in a leaf have the same class label. Records containing missing values are assigned to each tree, and the label of the related leaf is selected as the correct value. This method can correct only missing values and is limited to numeric and categorical data.

SiMI algorithm is an extended form of DMI algorithm [17]. In this method, the decision tree has been replaced by a decision forest. After building trees, this method starts to search for the intersection of different leaves. SiMI merges the smallest intersection with the intersection in which records are highly similar to each other. In addition, it maximizes the dependency between attributes when intersections are merged. In the last step, SiMI assigns each record with missing value to an intersection for finding a replacement value. As this algorithm is the extended form of DMI, it can correct only missing values of numeric and categorical types.

In [5], the researcher introduces bagging predictors as a cleaning method, in which multiple versions of the predictor are generated and at last, a final prediction is provided based on votes of all predictors. Multiple predictors are formed by bootstrap repeatedly on the training set. After that, they are used as new training sets. In this method, correction and detection phases cannot be separated. Therefore, the user is not able to employ another approach for the correction phase.

SCARE[§] algorithm combines machine learning algorithms with likelihood to correct erroneous data [32] using correct values. To use likelihood, two criteria have been employed for maximizing the likelihood benefit and minimizing the cost of changes. For this purpose, a classifier is formed for each attribute and the value of each dirty attribute is predicted. In the next step, a graph is constructed for each record where nodes are predicted values and edges represent the dependency between attributes. After constructing the graph, the node whose edges have the minimum weight is deleted until only one value of each attribute remains. In this method, it is supposed that the error has been identified and the approach only corrects erroneous records.

[†] Correlation Maximization-based Imputation Methods
[‡] Expectation Maximization
[§] SCalable Automatic REpairing

The algorithm provided in [26] uses predefined rules for the correction of an inconsistent data source. These rules consist of three components. The first and second ones build the left side of the rules and the third one builds the right side. The first part which is called an evidence pattern indicates attributes that are related to each other. The second part of the fixing rule consists of a negative patterns indicating wrong values of attributes. The last part is the actual value indicating the correct value of each wrong value. At first, each attribute provided in the evidence pattern is selected as a key. Then, they are saved together with their corresponding values in a list. In each record, keys which are saved in the list are searched. In the case of any correspondence, the second part of the rule is considered. If the second part of the rule is found in the record, it is corrected by the third part of the rule. This procedure is repeated for all records.

In [13], an approach has been introduced for correcting inconsistencies using rules. The rules existing in a data source are extracted using available algorithms. Then, the confidence of each rule is calculated. Any transaction violating these rules is suspected as errors. The algorithm assigns a score to each transaction based on the number of violated rules by it. That means each transaction which violates more rules, receives the higher score. Finally, transactions with their scores are displayed to users and able them to decide about the transaction based on the scores. As this method requires an expert in the correction phase, it is not appropriate for a large amount of data.

In [27, 28, 29], it is tried to identify attributes which are suspected of being noisy and, correct them by a polishing algorithm. This algorithm has two phases: prediction and adjustment. In the prediction phase, one algorithm is selected from the classification algorithms. Data is selected by a ten-fold cross-validation. In this phase, the value of the considered attribute is predicted for each record by each ten classifiers. If the original value of that record is inconsistent to the predicted value, the value of the record obtained from the prediction will be saved for correction in the next phase. In the adjustment phase, the ten-fold cross-validation is performed on attributes. Then, the incorrect value of each record which was identified in the previous phase is corrected based on the predicted value by ten classifiers. The correct value can be selected only from the values predicted in the previous phase. In this method, correction and detection phases cannot be separated and therefore, a user cannot employ another approach as the correction phase.

In [14], the constraints of databases have been employed for the correction of inconsistencies. At first, constraints existing in a data source are extracted, and then, records violating constraints are found. Records having inconsistencies are corrected by a greedy algorithm. As the extraction of constraints requires the interaction with a user, this method is not efficient for a large amount of data.

The method introduced in [15] is a rule-based approach for error detection. At first, all attributes are converted to binary form . After the binarization of attributes, the rules with the minimum support are extracted from a data source. Then, it is calculated how many times a rule is violated by the data of that source. Rules are deleted which violated more than a specified threshold level. In the next step, rules having a parent are deleted. That means, if there are two rules in the form of $x, y \to z$ and $x \to z$ in the rule set, then $x \to z$ is considered as the parent of $x, y \to z$. Therefore, $x, y \to z$ is deleted from the rule set. Thereafter, the number of violating rules by each record is calculated. Records which violated more than a threshold level are detected as erroneous records.

The proposed method in [7] repairs data using a knowledge base and a crowd powered. The table containing errors, the knowledge base, and the crowd are inputs in this method. At first, a table pattern is created for mapping data table to the knowledge base. Then, it chooses the best table pattern using a crowdsourcing. After that, data are categorized into three classes: (1) correct tuples that are identified using the knowledge base, (2) correct tuples identified using the knowledge base and the crowd, (3) dirty tuples identified using the knowledge base and the crowd. Finally, top k mappings are presented from the knowledge base as a correct value for erroneous data. This method uses an expert if there is not enough information for selecting k corrections.

A repairing method based on constraints has been proposed in [25]. The method finds a minimum data repair that satisfies at least one of the constraints which has variety more than other constraints. It employs both predicate insertion and deletion for repairing constraints. Compared to the related approach with the trust parameter that controls the portion of trusted data, it can corrects numeric data type.

The interaction between data correction and record matching has been studied as new problem in [9]. This approach indicates that data correction can help data matching effectively. This method presents a framework contains correction and matching to achieve a corrected data set based on constraints, verification rules, and

master data. This approach uses conditional functional dependencies and matching dependencies as constraints for detecting inconsistencies. This framework proposes three algorithms for error identification.

In [2] a novel data repairing approach is proposed based on constraints and ensemble learning. The proposed approach consists of two main steps. At first step, functional dependencies are extracted. Then, noisy records which violate functional dependencies more than a threshold are identified. After that, the repeated values are extracted from consistent records for each FD. The repeated values are used to detect noisy attributes. In the second step, ensemble learning model is used to correct noisy attributes.

A cost based model has been presented in [6] which data and constraints are compared equally. The model uses functional dependencies as constraints. The model proposes two separated algorithms for repairing data and constraints. Data repair algorithm looks for correct values for inconsistent records that have a minimum changes in original data. This model proposed two approaches to correct inconsistent records with a functional dependency. In the first approach, it finds a repair for two inconsistent records which have different values for the right side attribute of the functional dependency. After that, the attribute values in these records are set equal. In the second approach, it sets different values for the left side attribute of the functional dependency in these records. Furthermore, the algorithm for repairing inconsistencies in functional dependencies has two approaches, too. In the first approach, it searches an attribute which added to the functional dependency to remove the inconsistency. In the second approach, a set of attribute values is selected from records satisfying functional dependencies. These values determine the subset of consistent records with functional dependencies. Then a cost based model runs for selecting the best correction. The main goal of this model is to select a correction with minimum changes.

A functional dependency based integration system has been introduced in [24] for inconsistent data identification. This approach corrects data or functional dependencies for fixing inconsistencies. It uses three techniques for functional dependencies correction: adding attributes to a functional dependency, transforming a functional dependency to a conditional functional dependency, and redundancy identification in a functional dependency. The system consists of four components: violation detector, data repair generator, constraint repair generator and unified repair engine. At first, data and functional dependencies are sent to the violation detector module to identify inconsistent data. Then, inconsistent records with each functional dependency are detected. The inconsistent record is passed to the data repair generator and violated functional dependency is sent to the constraint repair generator. Data repair module compares records patterns and passes a set of corrections to the repair engine. Functional dependency correction module sends a set of corrections to the repair engine. Correction engine selects a repair with minimum cost.

Functional dependencies have been used widely for error detection. Different repairs can be employed for each identified inconsistency but just one repair has to be chosen as the final repair. To find the optimal repair, a cost and diversity function are used as two parameters. The algorithm provided in [12] uses both parameters as its objectives for the first time. To compute diversities, a distance function is used that computes dissimilarities between records. To calculate costs, the number of changes in the original data source is computed.

In [3], A method has been proposed for inconsistencies corrections using sampling from the repair space of conditional functional dependency. This method proposes more than one value for correcting each inconsistency. Firstly, this method detects clean cells of each tuple. Then, for each functional dependency it generates a set of consistent cells. Finally, it randomly selects an efficient correction using the sampling algorithm from the space of cardinality-set-minimal. This method can apply user determined constraints in addition to the functional dependencies and conditional functional dependencies for corrections. This method only corrects inconsistencies and does not delete records having them.

In [11] a repair diversification novel has been presented. The aim of this approach is to generate a set of repairs, such that these repairs minimize the cost and maximize the diversity. Actually, generated repairs are dissimilar with each other to prevent redundancy. In this approach, a user defines a parameter, in order to keep a balance between the cost and diversity.

In [20] a framework is introduced for data repairing by probabilistic inference engine. It unifies integrity constraints, external data, and quantitative statistics, to repair errors in structured data sets. For combination these methods, probability theory is used. A framework generates a probabilistic model to detect inconsistencies over records in the data set. Statistical learning and probabilistic inference engine are used in order to clean errors.

The methods mentioned above are different in terms of being automated or semi-automated, the ability in noise detection, and the type and number of errors they can detect. This paper aims to propose an automatic noise detection method in a data set which has different data types. In the next section, we detail our approach first and next, it compared with the approach presented in [15]. The purpose of selecting this method for comparison is that the method automatically and separately detects noises in the different types of data.

3. Proposed Method

In this section, we propose an automated noise detection method based on the *k*-means clustering and *k*-nearest neighbors classification. At first, each attribute (A_j) is temporarily removed from data and the *k*-means is applied. After that, in each cluster, the *k*-nearest neighbors is used in order to predict a value for A_j in each record. Noisy records are detected which A_j have incorrect values in them. The procedure is applied for all attributes. The proposed method is able to detect noise in the different data types. This method only detects noises and for correction them, any other approaches can be employed. The method consists of four phases explained in the following subsections. Algorithm 1 illustrates the following subsections 3.1 to 3.3 which form the main part of the proposed algorithm. In the next subsections, each phase of Algorithm 1 will be explained.

3.1. First Phase: Clustering

Suppose that the input data set (D) has been defined on a set of attributes $\{A_1, A_2, \cdots, A_m\}$. In this paper, D_{ij} refers to j^{th} attribute of i^{th} record in D. In this phase, the *k*-means clustering is applied for each of *m* attributes. That means A_j is temporarily taken from the data source for each A_j attribute, and the *k*-means clustering is applied on the other attributes (all attributes except A_j). To select *k*, one of the cluster validity indices [1] can be employed. In this paper, Silhouette Index has been used to determine the number of clusters [21]. After each clustering, the second, third, and fourth phases are implemented for each attribute. Specifically, in this phase, one attribute (A_j) is removed from the data source in order to detect records having incorrect values for their j^{th} attribute. In this paper, it is supposed that there is not any master data or primary knowledge for examined data set to use for detecting noisy fields. After that, the *k*-means clustering is used in order to find similar records. In the data source, the other attributes may be incorrect in records in addition to A_j, so some correct attributes are detected as noises. To solve this problem, the proposed approach is iterated (fourth phase), because the *k*-means has different outputs in iterations on the same data source. Lines 8 and 9 of Algorithm 1 show first phase of the algorithm. In line 8, the considered attribute will be deleted from the data. In line 9, the *k*-means clustering is carried out for other attributes.

3.2. Second Phase: Comparison

After each clustering, nearest neighbors are found for each record (*r*) among records in the same cluster with *r*. After that, A_j is put as target for each record and a value is predicted for it in the record *r* based on majority of votes or average value using nearest neighbors. That means the *k*-nearest neighbors is applied for all records in all clusters. Suppose that a record *r* is in the cluster *i*. To applies the *k*-nearest neighbors, $k^{'}$ records, which are in the cluster *i* and have the least distance from the record *r*, are selected for the record *r*. The value of $k^{'}$ is an experimental value. If the attribute A_j is a numeric, the average of A_j in $k^{'}$ neighbors is predicted for the value of A_j in the record r. However, if A_j is of ordinal or nominal type, the most frequent value for A_j in this $k^{'}$ neighbors is predicted. The predicted value is called β. Specifically, these values are used in next phase in order to detect in which record, A_j has incorrect value. In fact, the nearest records are assumed as correct data and used to predict a value for A_j in each record.

Lines 10 to 22, and 27 to 32 of Algorithm 1 show this phase of the algorithm. In lines 10 to 16, the distance of the record *r* in the cluster *i* from other records is calculated and the $k^{'}$ records which have the least distance from the record *r* in the cluster *i* are selected. Lines 17 to 22 and 27 to 32 estimate values for numeric, ordinal, and nominal attributes, respectively. To calculate the distance in this method, Euclidean distance has been employed.

3.3. Third Phase: Error Detection

If the attribute A_j is numeric and the difference between its value attribute in record r and β is higher than a threshold level (ϕ), that record detected as a noisy one. Moreover, if the attribute has ordinal or nominal type and the value of A_j in record r is in contradiction with the value of β, A_j is detected as the noise in record r. In fact, the proposed algorithm detects records having incorrect A_j in this phase. In lines 23 to 25 of Algorithm 1, the noise detection of numeric attributes and in lines 33 to 36 the noise detection of nominal and ordinal attributes have been done, respectively. It is noteworthy that, the *k*-nearest neighbor predict a value close to real value, so the predicted value is not sufficient in order to use for correction noise. In the other word, the predicted value by *k*-nearest neighbor is the initial estimation, and suggested to use another approaches [17, 16, 32] to correct the detected noise.

3.4. Fourth Phase: Error Reduction

This algorithm detects not only noisy attributes but also some correct ones as noisy attributes. To minimize the number of correct attributes identified as a noisy one, the steps 3-1 to 3-3 are run at least 2 times and at most 5 times with $k^{'}$ neighbors to increase the efficiency of the algorithm. The number of iterations has been obtained from the experiments conducted on the different data sets. Taken into account that there is a difference between the approach for numeric attributes and the approach of nominal and ordinal attributes, two different approaches have been proposed for reducing errors. For numeric attributes, the iteration showing more errors will be selected. This iteration is called n_1 and it is compared with other iterations. Suppose that n is the number of iterations, if an erroneous attribute which is in n_1 occurs in less than $n-1$ iterations, its value in the record will be considered as correct value.

As regards nominal and ordinal attributes, just like numeric attributes, the iteration is selected which has most noisy attributes. It is called n_1 and compared with the other iterations. The elements existing in less than $n-1$ iterations are deleted and the other ones are saved in a temporary memory. The steps 3-1 to 3-3 are run at least 2 and at most 5 times with $k^{'}-1$ neighbors. However, the results of iterations are compared at this time with the elements existing in the temporary memory. Any element of the memory which exists in results of *n* iterations is detected as the final error.

3.5. Time Complexity

In this section, the time complexity of the proposed algorithm is calculated. Suppose that a data set D includes s records and m attributes and also k is the cluster number of the *k*means algorithm. In line 7, there is a general loop, which runs lines 8-38 of the algorithm *m* times. At first, the k-means algorithm is applied to cluster the data set. The order of the k-means algorithm is *O(kst)* and t is the iteration number of the main body of the *k*-means algorithm. In line 12, there is a loop which iterates *s* times. Inside this loop and in lines 12-14, the distance of each record r from other records which have the same cluster is calculated. If $k=s$, these lines will run once, and if $k=1$, they will iterate *s* times.

In line 16, the obtained distances are put in a ascending order. If the first condition is met, the order is $O(1)$ and if the second condition is fulfilled, the order is $O(s\,logs)$. In lines 19-21 and 29-31, there is a loop, which is iterated $k^{'}$ times. If the first condition is met, $k^{'}=1$ and the loop is not iterated. However, if the second condition is met $k^{'}=s$ and this loop is iterated *s* times. Therefore, the order of this algorithm is $O(ks^2logs)$ at the worst case. As the procedure of error detection and correction is considered as the preprocessing phase and has no effect on the main data processing, the order of correction algorithm does not affect the main processing step.

4. Experiments

In this section, five different data sets taken from the UCI Machine Learning Database, are used to assess the proposed method performance. Table 1 shows a brief summary of these data sets. These five data sets are correct and free from any error. To implement this method, MATLAB R2014a has been employed. The first data set is about

Algorithm 1 The pseudo-code of the steps 3-1 to 3-4 of the proposed algorithm

Input parameters
D: A data set defined on schema $\{A_1, A_2, \cdots, A_m\}$ with *m* attributes and *s* records
k: Number of the *k*-means cluster
ϕ : Threshold for difference between the predicted value and the considered value
Output: A set of erroneous records
Function
for all attributes A_j in D **do**
  $D^{'} \leftarrow$ remove A_j from D
  $RecordsClusterLabel \leftarrow k-$means$(D^{'}, k)$
  for all records *r* in $D^{'}$ **do**
    $ClusterLabel \leftarrow$ find cluster of record r from $RecordsClusterLabel$
    for all records *r* which cluster of it is *ClusterLabel* **do**
      $dis(r) \leftarrow$ euclidean distance between records in $ClusterLabel$ and r
    $Indexs \leftarrow$ sort records $\in ClusterLabel$ according to dis in a ascending order
    $Indexs \leftarrow$ select top $k^{'}$ of $Indexs$
    if A_j is numeric **then**
      $sum \leftarrow 0$
      for all *i* in *Indexs* **do**
        $sum \leftarrow sum + D^{'}_{ij}$
      $sum \leftarrow sum/k^{'}$
      if $(sum - D^{'}_{ij}) > \phi$ **then**
        $out \leftarrow out \cup r$
    if A_j is nominal or ordinal **then**
      $sum \leftarrow Null$
      for all *i* in *Indexs* **do**
        $sum \leftarrow sum \cup D^{'}_{ij}$
      $sum \leftarrow$ find the most repeated index in sum
      if $(sum \neq D^{'}_{rj})$ **then**
        $out \leftarrow out \cup r$
    return out
ENDFunction

wholesale customers [¶] and consists of eight attributes. The second data set is about user knowledge modeling [‖] and consists of six attributes. The third data set is about the income of people based on census [**] and consists of fifteen attributes. The fourth data set is about the Indonesia contraceptive prevalence [††] and consists of nine attributes. The fifth data set is about predicting the cellular localization sites of proteins [‡‡] and consist of nine attributes.

In this paper, the performance of the proposed algorithm is evaluated by five criteria. Before introducing the criteria, the parameters used by these criteria are introduced. N and P denote the number of erroneous fields and the number of correct fields, respectively. TN and TP are respectively the numbers of erroneous and correct fields which have been labeled correctly by the algorithm. FP and FN are the numbers of erroneous and correct fields which have been labeled wrongly by the algorithm, respectively. In the following we introduce the criteria. By the first criterion shown in the equation (1), false alarm rate has been calculated. This rate is the number of errors

[¶] https://archive.ics.uci.edu/ml/datasets/Wholesale+customers
[‖] https://archive.ics.uci.edu/ml/datasets/User+Knowledge+Modeling
[**] https://archive.ics.uci.edu/ml/datasets/Adult
[††] https://archive.ics.uci.edu/ml/datasets/Contraceptive+Method+Choice
[‡‡] https://archive.ics.uci.edu/ml/datasets/Yeast

which have not been detected divided by the total number of errors. By the second criterion shown in the equation (2), the error rate is calculated. For this purpose, the total number of the errors which have not been detected and the number of the fields which have been correctly detected as erroneous fields are divided by the total number of all records. An equation (3) calculates the true negative rate. This rate equals the number of the detected errors divided by the total number of the detected errors plus those errors which have not been detected. By the equation (4), the recall is calculated. For this purpose, the number of correct attributes which have been detected correctly is divided by the total number of the correct attributes that have been detected correctly plus the number of the correct attributes that have been detected as erroneous ones. The equation (5) calculates the precision. This rate equals to the number of correct attributes which have been detected correctly divided by the total number of the correct attributes that have been detected correctly plus the number of errors which have not been detected. The high value of the true negative rate, precision, and recall, and the low value of the false alarm rate and error rate in the experiments show the high efficiency of the algorithm.

$$falsealarmrate = \frac{FP}{FP + TN} \tag{1}$$

$$errorrate = \frac{FP + FN}{N + P} \tag{2}$$

$$truenegativerate = \frac{TN}{TN + FP} \tag{3}$$

$$recall = \frac{TP}{TP + FN} \tag{4}$$

$$precision = \frac{TP}{TP + FP} \tag{5}$$

In each data set which were free of error, 5%, 10%, 15%, and 20% noises are created and the performance was calculated using the above mentioned criteria. Random function is used to create noises in nominal and ordinal attributes. This function generates random numbers from the domain of attributes. For numeric attributes, equation (6) is used to create noises. A Rand() in equation (6), generate random number between 0 and 1. In figures 1 to 5 the performance of the proposed method is shown in five data sets using mentioned criteria. As shown in these figures, in all noise rates, the true negative rate, recall, false alarm rate, error rate, and precision are averagely 92%, 96%, 8%, 5%, and 98.5% respectively. Table 2 shows the performance of the proposed method with the noise rate equal to 20% in each data set and for different iterations. The true negative is more than 89% in all iterations and the false alarm rate is averagely 9%. In the iteration 1, in all data sets, the maximum value of the error rate is 17%, and after increasing the number of iterations, this value decreased to 9%. The best balance between parameters for the two first data sets, Wholesale Customer and User Knowledge Modeling, has been achieved in the iteration 3, for Adult and Contraceptive Method Choice in iteration 5, and for Yeast in iteration 4.

$$A_j = min(A_j) + abs((max(A_j) - min(A_j)) \times Rand()) \tag{6}$$

In the numeric attributes, the value of ϕ has been introduced as the difference between the predicted value from the $k-$nearest neighbors and the actual value of the attribute in the considered record. To obtain the best value of ϕ, 10% of noises were created in a numeric attribute, and the five criteria were considered in terms of different values of ϕ. Table 3 shows different values of the parameter ϕ. As shown, the low value of ϕ causes the true negative rate to be 100%. However, it increases the error rate to the maximum rate equal to 25%. The increase in the value of ϕ reduces the error rate dramatically. In addition, the true negative rate decreases from 100% to a value higher than 89%. The value of ϕ to achieve the best balance between criteria for the Wholesale Customer data set is 2350, for the User Knowledge Modeling data set is 0.285, for the Adult data set is 9, for the Contraceptive Method Choice data set is 0.31, and for the Yeast data set is 0.13. Any higher value of ϕ reduces true negative rate. Thus, these values are the best ones for ϕ in the data sets. According to the experiments, this best value is something between the mean and covariance value of each attribute.

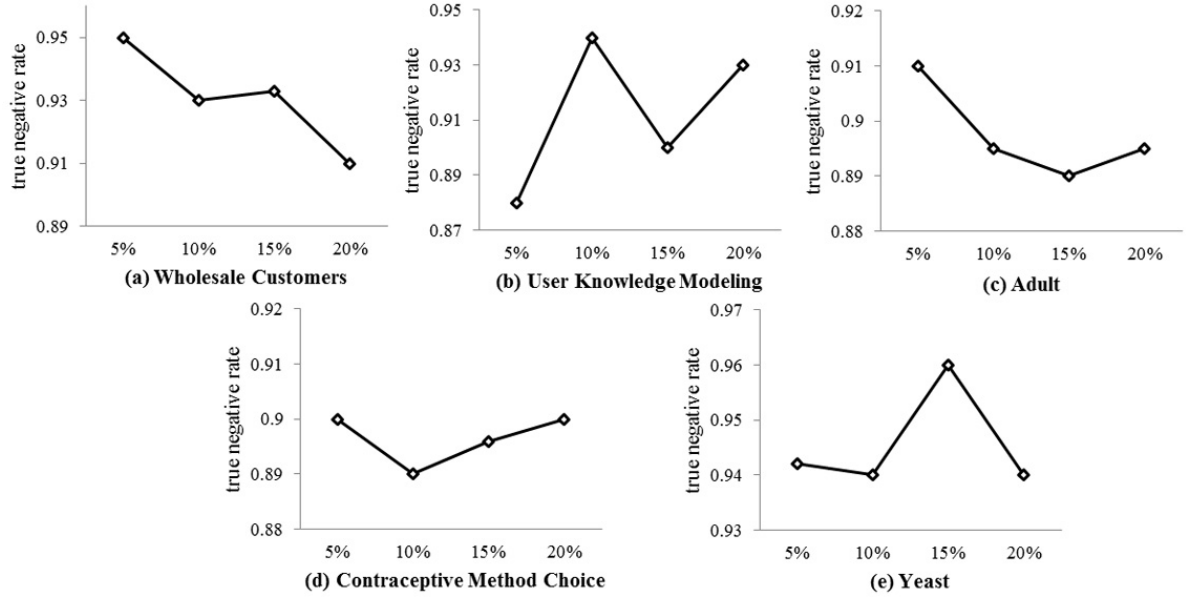

Figure 1. The values of true negative rate in five data sets

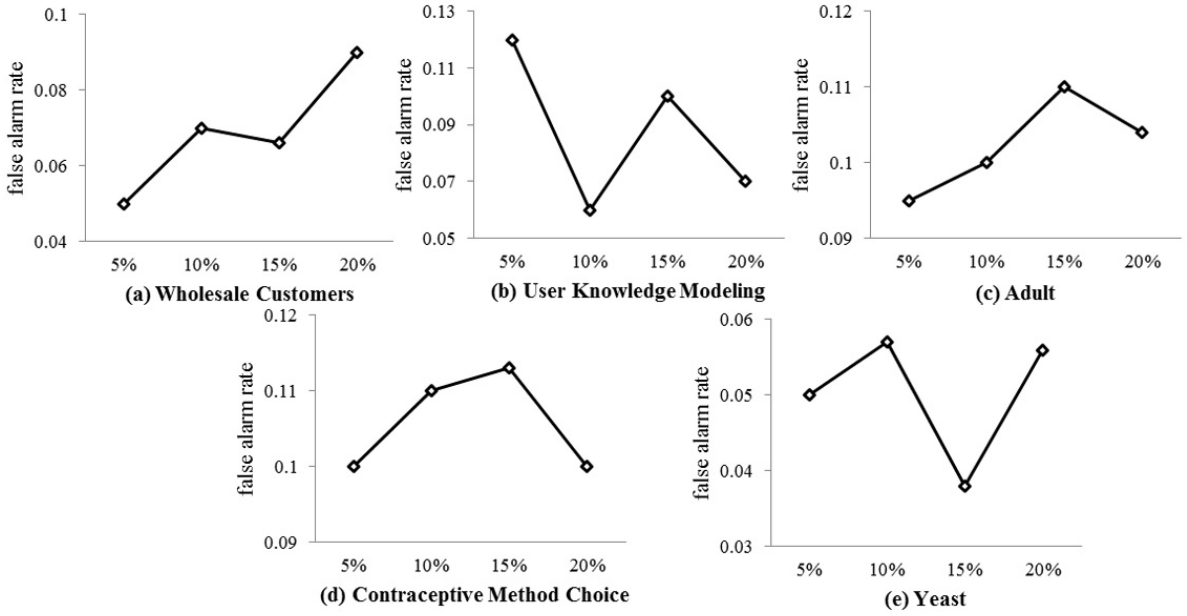

Figure 2. The values of false alarm rate in five data sets

In Table 4, from the Wholesale Customer, Adult, and Yeast data sets a numeric attribute, and from User Knowledge Modeling, and Contraceptive Method Choice data sets an ordinal attribute has been selected. This table tests a number of neighbors to find the best number of neighbors. As shown in Table 4, at first, the true negative rate, precision, and recall are low and the false alarm rate and error rate are high. The increase in the number of neighbors, increases the true negative rate, precision, and recall and decreases false alarm rate and error rate. The best balance between criteria for the Wholesale Customer data set is achieved in the 8 neighbors, for the User Knowledge Modeling data set in the 4 neighbors, for the Adult data set in the 20 neighbors, for the Contraceptive

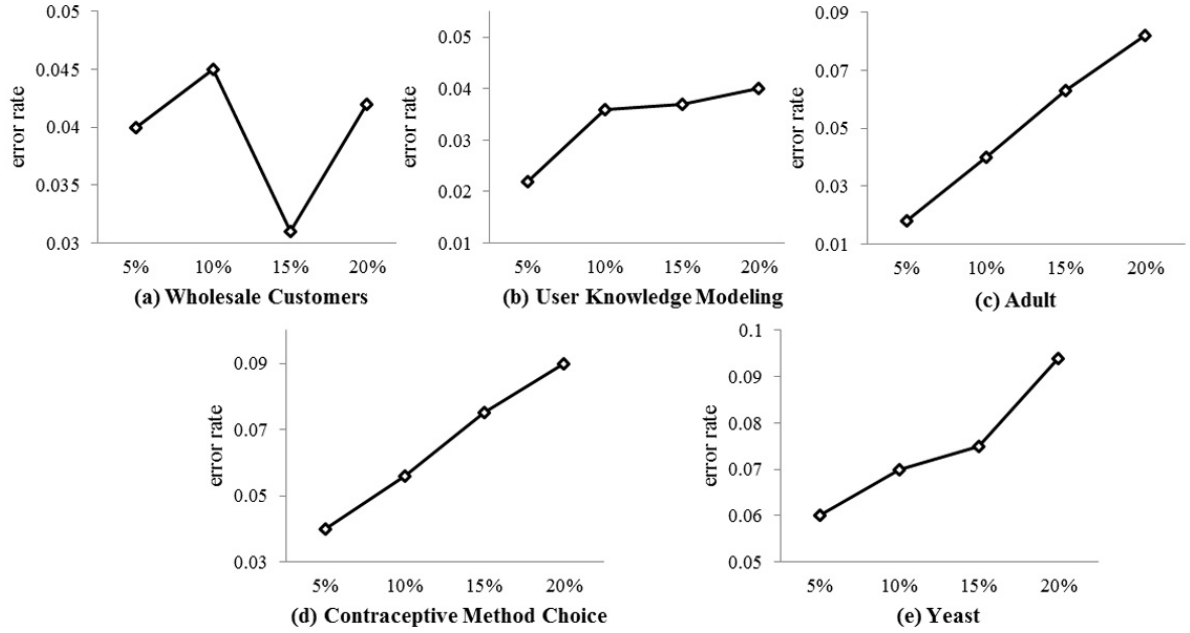

Figure 3. The values of error rate in five data sets

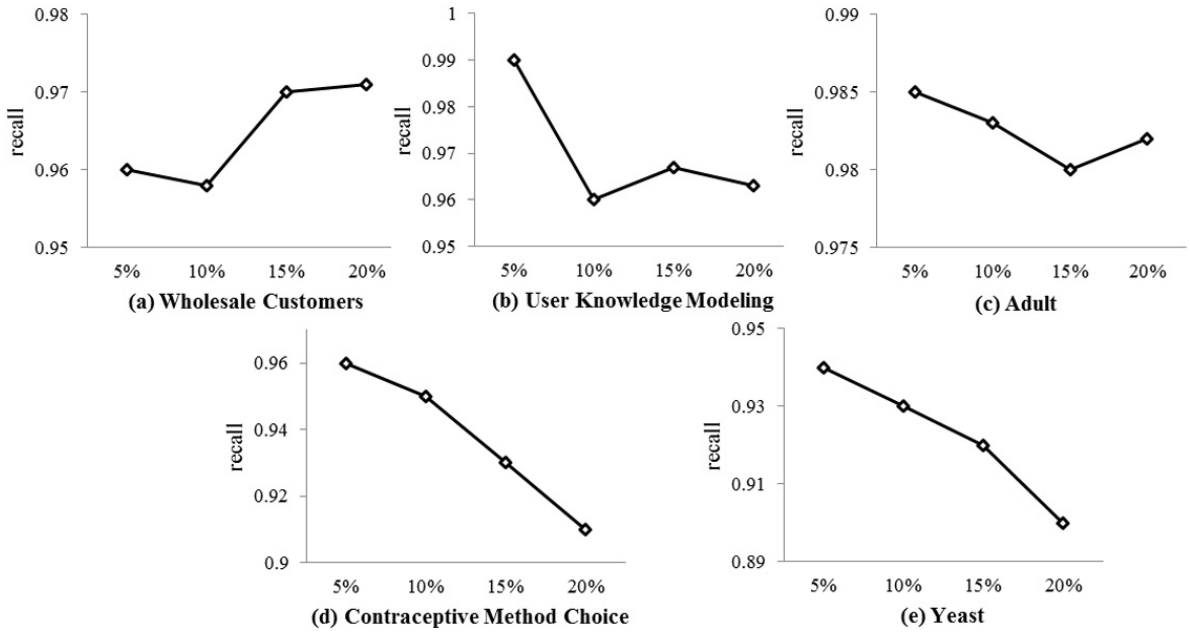

Figure 4. The values of recall in five data sets

Method Choice in 22, and for the Yeast data set in 30. In these points, the true negative rate, precision, and recall reach its maximum values and the false alarm rate and error rate reduce considerably.

In Figure 6, the method proposed in this paper has been compared with the automated noise detection method using rules [15]. The rule based approach has been selected for comparison for three reasons: firstly, it has only detection phase, secondly, it is able to detect noises in all types of data, and finally, it detects noisy fields automatically without user interaction. To implement the automated noise detection method using association rules

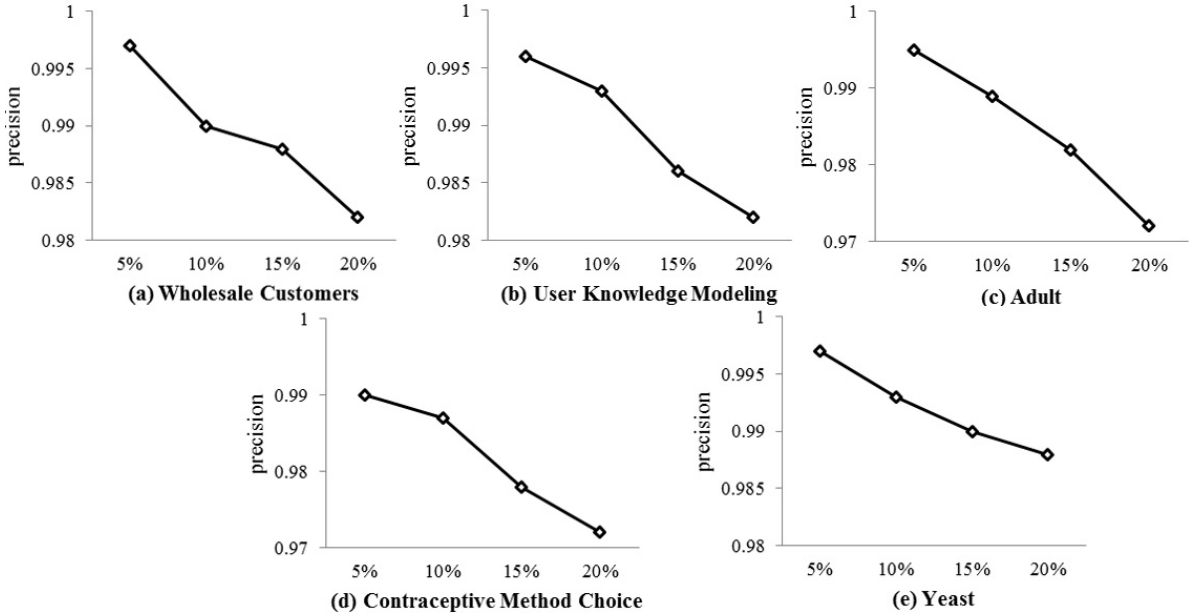

Figure 5. The values of precision in five data sets

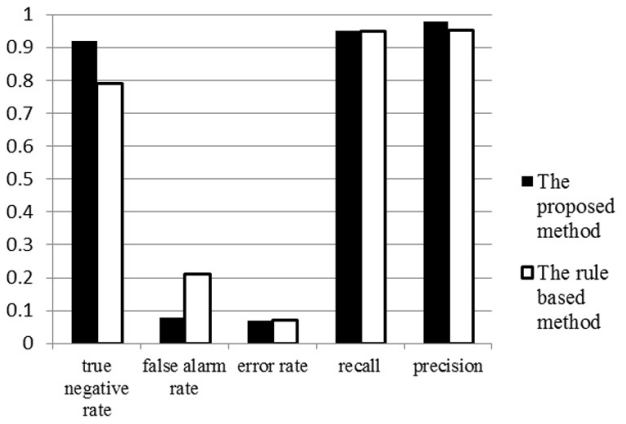

Figure 6. The comparison between the proposed method and the rule based method

[15], C# language and Visual Studio 2010 are employed. An average performance of the proposed algorithm is compared with an average performance of the automated noise detection method using rules. In all data sets 20% of noises are created.

As shown in Figure 6, the value of the recall in the proposed method and the automated noise detection method using rules is equal, but the rest criteria in the proposed method in comparison with the automated noise detection method using rules have better results. In the proposed method, the true negative rate and precision increase averagely by 13% and 2.7% respectively from those of the automated noise detection using rules. In addition, false alarm rate and error rate decrease by 13% and 0.1% respectively from those of the automated noise detection using rules. The experiments indicate that the proposed method is more efficient than the rule based method [15].

Table 1. Data sets at a glance

Data set	# of Instances	# of nominal Attributes	# of numeric Attributes	# of ordinal Attributes
Wholesale Customers	440	2	6	—
User Knowledge Modeling	400	—	5	1
Adult	48842	8	6	1
Contraceptive Method Choice	1437	3	2	4
Yeast	1484	1	8	—

Table 2. Analysis of a suitable number for the iteration in five data sets

Data set	# of Iteration	True negative rate	False alarm rate	Error rate	recall	precision
Wholesale Customers	1	0.95	0.042	0.084	0.9	0.989
	2	0.94	0.056	0.07	0.925	0.986
	3	**0.91**	**0.084**	**0.042**	**0.968**	**0.981**
	4	0.92	0.079	0.044	0.966	0.98
	5	0.915	0.08	0.042	0.968	0.98
User Knowledge Modeling	1	0.875	0.125	0.062	0.95	0.97
	2	0.89	0.11	0.056	0.956	0.975
	3	**0.93**	**0.07**	**0.04**	**0.966**	**0.986**
	4	0.9	0.1	0.049	0.959	0.979
	5	0.84	0.16	0.046	0.976	0.966
Adult	1	0.93	0.068	0.12	0.867	0.98
	2	0.94	0.056	0.11	0.876	0.984
	3	0.92	0.07	0.1	0.89	0.98
	4	0.9	0.1	0.09	0.9	0.975
	5	**0.895**	**0.104**	**0.081**	**0.92**	**0.972**
Contraceptive Method Choice	1	0.94	0.06	0.15	0.98	0.827
	2	0.935	0.065	0.13	0.981	0.838
	3	0.93	0.07	0.11	0.98	0.88
	4	0.91	0.09	0.1	0.975	0.87
	5	**0.9**	**0.1**	**0.091**	**0.972**	**0.91**
Yeast	1	0.9	0.1	0.17	0.83	0.965
	2	0.92	0.08	0.14	0.87	0.975
	3	0.935	0.065	0.11	0.89	0.97
	4	**0.94**	**0.056**	**0.094**	**0.9**	**0.988**
	5	0.938	0.062	0.098	0.905	0.98

Table 3. Analysis of a suitable value for ϕ in five data sets

Data set	Value of ϕ	True negative rate	False alarm rate	Error rate	recall	precision
Wholesale Customers	1000	1	0	0.25	0.72	1
	1500	1	0	0.014	0.84	1
	2000	0.98	0.02	0.037	0.96	0.997
	2350	**0.95**	**0.05**	**0.02**	**0.98**	**0.994**
	2500	0.9	0.1	0.017	0.99	0.988
	3000	0.65	0.35	0.035	1	0.961
User Knowledge Modeling	0.15	1	0	0.22	0.78	1
	0.2	1	0	0.189	0.81	1
	0.25	1	0	0.060	0.94	1
	0.285	**0.93**	**0.07**	**0.027**	**0.98**	**0.97**
	0.3	0.87	0.13	0.033	0.977	0.96
Adult	1	1	0	0.005	0.994	1
	3	1	0	0.0042	0.995	1
	6	0.92	0.08	0.0042	0.995	0.999
	9	**0.89**	**0.11**	**0.004**	**0.996**	**0.998**
	12	0.87	0.13	0.0038	0.996	0.992
	15	0.78	0.22	0.0038	0.997	0.998
Contraceptive Method Choice	0.25	1	0	0.21	0.79	1
	0.27	0.94	0.06	0.18	0.82	0.999
	0.29	0.94	0.06	0.09	0.91	0.999
	0.31	**0.89**	**0.11**	**0.056**	**0.94**	**0.997**
	0.33	0.87	0.13	0.052	0.949	0.995
	0.35	0.83	0.17	0.05	0.95	0.99
Yeast	0.09	1	0	0.17	0.83	0.965
	0.11	1	0	0.14	0.87	0.975
	0.13	**0.93**	**0.07**	**0.11**	**0.89**	**0.97**
	0.15	0.78	0.21	0.094	0.9	0.988
	0.17	0.72	0.28	0.098	0.905	0.98

5. Conclusion

Data accuracy is considered as an important dimension in data quality. The decision making based on incorrect data can impose high costs and failure on organizations. The data cleaning process consists of two phases: error detection and error correction. The cleaning process detects inconsistencies, missing values, and duplicates; and corrects the detected errors. Considering the high volume of data, the interaction with the user is impossible during error detection. Therefore an automatic approach has been proposed for error detection in this paper. The proposed method is based on the $k-$means clustering. In this approach, clustering was carried out for each attribute, and then in each cluster, the $k-$nearest neighbors was applied. This approach can detect errors in different data types. In addition, as attributes are considered separately, it can detect several erroneous attributes in a record. According to the experiments, the true negative rate of this method is averagely equal to 92%. Moreover, the true negative rate of the proposed algorithm is 13% more than that of the similar method.

Table 4. Analysis of a suitable value for the number of neighbors in five data sets

Data set	# of neighbor	True negative rate	False alarm rate	Error rate	recall	precision
Wholesale Customers	2	0.15	0.84	0.098	0.983	0.91
	4	0.52	0.48	0.052	0.997	0.95
	6	0.61	0.39	0.039	0.997	0.96
	8	**0.93**	**0.05**	**0.12**	**0.99**	**0.994**
	10	0.91	0.09	0.021	0.988	0.988
	12	0.92	0.07	0.018	0.988	0.99
	14	0.9	0.1	0.019	0.988	0.988
User Knowledge Modeling	2	0.51	0.49	0.138	0.9	0.94
	3	0.78	0.23	0.061	0.96	0.968
	4	**0.93**	**0.07**	**0.027**	**0.975**	**0.993**
	5	0.9	0.1	0.031	0.978	0.987
	6	0.9	0.1	0.035	0.962	0.987
Adult	10	0.63	0.36	0.011	0.993	0.994
	15	0.85	0.15	0.007	0.994	0.997
	20	**0.95**	**0.05**	**0.003**	**0.997**	**0.997**
	25	0.89	0.11	0.004	0.997	0.996
	30	0.88	0.12	0.005	0.997	0.996
Contraceptive Method Choice	18	0.8	0.2	0.11	0.89	0.991
	20	0.94	0.06	0.09	0.9	0.999
	22	**1**	**0**	**0.06**	**0.94**	**1**
	24	0.94	0.06	0.062	0.938	0.999
	26	0.94	0.06	0.064	0.93	0.999
Yeast	15	1	0	0.23	0.77	1
	20	1	0	0.12	0.88	1
	25	1	0	0.03	0.97	1
	30	**0.93**	**0.07**	**0.011**	**0.96**	**0.999**
	35	0.79	0.21	0.013	0.988	0.997
	40	0.72	0.28	0.015	0.986	0.996

REFERENCES

1. O. Arbelaitz, I. Gurrutxaga, J. Muguerza, J. M. Prez, and I. Perona. An extensive comparative study of cluster validity indices. *Pattern Recognition*, 46(1):243 – 256, 2013.
2. M. Ataeyan and N. Daneshpour. A novel data repairing approach based on constraints and ensemble learning. *Expert Systems with Applications*, 159:113511, 2020.
3. G. Beskales, I. F. Ilyas, L. Golab, and A. Galiullin. Sampling from repairs of conditional functional dependency violations. *The VLDB Journal*, 23(1):103–128, Feb. 2014.
4. L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.
5. S. Brüggemann. *Rule Mining for Automatic Ontology Based Data Cleaning*, chapter Progress in WWW Research and Development: 10th Asia-Pacific Web Conference, APWeb 2008, Shenyang, China, April 26-28, 2008. Proceedings, pages 522–527. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

6. F. Chiang and S. Sitaramachandran. Unifying data and constraint repairs. *J. Data and Information Quality*, 7(3):9:1–9:26, Aug. 2016.
7. X. Chu, J. Morcos, I. F. Ilyas, M. Ouzzani, P. Papotti, N. Tang, and Y. Ye. Katara: A data cleaning system powered by knowledge bases and crowdsourcing. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, SIGMOD '15, pages 1247–1261, New York, NY, USA, 2015. ACM.
8. W. Fan. Dependencies revisited for improving data quality. In *Proceedings of the Twenty-seventh ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '08, pages 159–170, New York, NY, USA, 2008. ACM.
9. W. Fan, S. Ma, N. Tang, and W. Yu. Interaction between record matching and data repairing. *J. Data and Information Quality*, 4(4):16:1–16:38, May 2014.
10. Y. Gao, C. Ge, X. Miao, H. Wang, B. Yao, and Q. Li. A hybrid data cleaning framework using markov logic networks. *CoRR*, abs/1903.05826, 2019.
11. C. He, Z. Tan, Q. Chen, and C. Sha. Repair diversification: A new approach for data repairing. *Information Sciences*, 346:90 – 105, 2016.
12. C. He, Z. Tan, Q. Chen, C. Sha, Z. Wang, and W. Wang. *Repair Diversification for Functional Dependency Violations*, chapter Database Systems for Advanced Applications: 19th International Conference, DASFAA 2014, Bali, Indonesia, April 21-24, 2014. Proceedings, Part II, pages 468–482. Springer International Publishing, 2014.
13. J. Hipp, U. Gntzer, and U. Grimmer. Data quality mining – making a virtue of necessity. In *PROCEEDINGS OF THE 6TH ACM SIGMOD WORKSHOP ON RESEARCH ISSUES IN DATA MINING AND KNOWLEDGE DISCOVERY*, pages 52–57, 2001.
14. A. Lopatenko and L. Bravo. Efficient approximation algorithms for repairing inconsistent databases. In *2007 IEEE 23rd International Conference on Data Engineering*, pages 216–225, April 2007.
15. W. A. Malik and A. Unwin. Automated error detection using association rules. *Intell. Data Anal.*, 15(5):749–761, Sept. 2011.
16. G. Rahman and Z. Islam. A decision tree-based missing value imputation technique for data pre-processing. In *Proceedings of the Ninth Australasian Data Mining Conference - Volume 121*, AusDM '11, pages 41–50, Darlinghurst, Australia, Australia, 2011. Australian Computer Society, Inc.
17. M. G. Rahman and M. Z. Islam. Missing value imputation using decision trees and decision forests by splitting and merging records: Two novel techniques. *Knowledge-Based Systems*, 53:51 – 65, 2013.
18. J. Rammelaere and F. Geerts. Explaining repaired data with cfds. *Proc. VLDB Endow.*, 11(11):1387–1399, July 2018.
19. J. Rammelaere and F. Geerts. Cleaning data with forbidden itemsets. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–1, 2019.
20. T. Rekatsinas, X. Chu, I. F. Ilyas, and C. Ré. Holoclean: Holistic data repairs with probabilistic inference. *Proc. VLDB Endow.*, 10(11):1190–1201, Aug. 2017.
21. P. J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53 – 65, 1987.
22. A. M. Sefidian and N. Daneshpour. Missing value imputation using a novel grey based fuzzy c-means, mutual information based feature selection, and regression model. *Expert Systems with Applications*, 115:68 – 94, 2019.
23. A. M. Sefidian and N. Daneshpour. Estimating missing data using novel correlation maximization based methods. *Applied Soft Computing*, 91:106249, 2020.
24. J. Segeren, D. Gairola, and F. Chiang. Condor: A system for constraint discovery and repair. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, CIKM '14, pages 2087–2089, New York, NY, USA, 2014. ACM.
25. S. Song, H. Zhu, and J. Wang. Constraint-variance tolerant data repairing. In *Proceedings of the 2016 International Conference on Management of Data*, SIGMOD '16, pages 877–892, New York, NY, USA, 2016. ACM.
26. N. Tang. *Big Data Cleaning*, chapter Web Technologies and Applications: 16th Asia-Pacific Web Conference, APWeb 2014, Changsha, China, September 5-7, 2014. Proceedings, pages 13–24. Springer International Publishing, 2014.
27. C.-M. Teng. Correcting noisy data. In *Proceedings of the Sixteenth International Conference on Machine Learning*, ICML '99, pages 239–248, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.
28. C.-M. Teng. A comparison of noise handling techniques. In *Proceedings of the Fourteenth International Florida Artificial Intelligence Research Society Conference*, pages 269–273. AAAI Press, 2001.
29. C. M. Teng. Polishing blemishes: issues in data correction. *IEEE Intelligent Systems*, 19(2):34–39, Mar 2004.
30. P. H. Williams, C. R. Margules, and D. W. Hilbert. Data requirements and data sources for biodiversity priority area selection. *Journal of Biosciences*, 27(4):327–338, 2002.
31. J. Y. Xiang, S. Lee, and J. K. Kim. Data quality and firm performance: empirical evidence from the korean financial industry. *Information Technology and Management*, 14(1):59–65, Mar 2013.
32. M. Yakout, L. Berti-Équille, and A. K. Elmagarmid. Don't be scared: Use scalable automatic repairing with maximal likelihood and bounded changes. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, SIGMOD '13, pages 553–564, New York, NY, USA, 2013. ACM.