Noisy values detection and correction of traffic accident data

Noisy Values Detection and Correction of Traffic Accident Data

Rupam Deb, Alan Wee-Chung Liew

School of Information and Communication Technology,

Gold Coast campus, Griffith University, QLD 4222, Australia

rupam.deb@griffithuni.edu.au, a.liew@griffith.edu.au

Abstract

Death, injury, and disability from road traffic crashes continue to be a major global public health problem. Therefore, methods to reduce accident severity are of significant interest to traffic agencies and the public at large. Noisy data in the traffic accident dataset obscure the discovery of important factors and mislead conclusions. Identifying and correcting noisy values is an important goal of data cleansing and preprocessing. This paper proposes a new algorithm called NoiseCleaner to identify and correct noisy categorical attributes values in large traffic accident datasets. We evaluate our algorithm using four publicly available traffic accident datasets from Australia and United States, namely, two road crash datasets from the Queensland Government data depository (data.qld.gov.au) and two datasets from the New York's open data portal (data.ny.gov). We compare our technique with several existing state-of-the-art methods and show that our algorithm performs significantly better than the existing algorithms.

Keywords: Data cleansing; Noisy value detection; Road traffic accident; Data preprocessing; Categorical data

1. Introduction

Real world data are often corrupted by noise and are generally incomplete, noisy, and inconsistent [8-11]. This occurs due to errors in data collection, storage and processing. Approximately 5% or more noisy data could be introduced to a dataset if an organization does not take extreme care during data collection [15]. The detection of anomalous records is especially important in view of today's massive datasets, where emphasis is often placed on the volume rather than the quality of the data. Noisy data can adversely affect the results of any downstream data analysis. The presence of noise reduces the quality of the models learned from the data, and impairs their predictive or descriptive performance. Moreover, these models would become overly complex in order to accommodate such noise.

In order to improve the quality of the data, raw data is often preprocessed before being analysed. Data preparation or preprocessing is one of the most critical steps in a data mining process which deals with the preparation and transformation of the initial dataset to remove noise and inconsistency in the data. Data preprocessing includes imputation of missing values, smoothing out noisy data, identification of incorrect data, and correction of inconsistent data, and could form a significant part of the total data mining effort.

The high growth of the number of vehicles has led to roads with higher traffic density. The immediate effect of this situation is the dramatic increase in traffic accidents on the road, which has become a serious problem for both developed and developing countries. A number of factors contribute to the risk of collision, including vehicle design, speed of operation, road design, road environment, driver skill or impairment, and driver behaviour. Worldwide, motor vehicle collisions lead to death and disability as well as financial costs to both society and the individuals involved. Road safety, which is mainly affected by road accident, is said to be one of the major health concerns. Nowadays, a large amount of traffic data has been collected with the advancement in sensor technologies. Using data mining technology, we can uncover patterns of traffic activities and factors that lead to accidents, which can then be used in various decision making process [1, 7, 18].

In the datasets we are concerned with, the attributes in a record are mostly categorical in nature. It is very hard to make categorical datasets using modern intelligent technology. For example, it is quite impossible to produce real world traffic accident dataset using sensor technology. Both human and machine participation need to create the categorical datasets. As human participation includes generating the categorical datasets therefore typo, missing values, and noisy values are very common. Although, many algorithms have been proposed for noisy data detection and correction, most of these algorithms deal only with numerical data. Detecting and correcting erroneous attribute values in categorical datasets is still a challenging problem and any attempt to numericalize a categorical value would introduce unwanted biases that could negatively affect subsequent data analysis. In this paper, we propose a novel method called NoiseCleaner for detecting noisy attributes values and predicting their correct values on traffic crash datasets. Most of the attributes of these records are categorical, ordinal or interval in nature. Categorical (sometimes called nominal) variables have values that have no natural ordering (e.g., airbag conditions: ruptured, cut, torn); ordinal variables do have a natural ranking order (e.g., day of the week); and interval variables are created from intervals on a contiguous scale (e.g., age group 13-19). Our contribution is novel in several aspects: we propose a probability measure inferred from statistics within the dataset to identify potential noisy attribute values, and we use both the direct and transitive similarity between attribute values to infer the correct value to be replaced with. This paper is organized as follows: in section 2, a literature review of related work is presented. Our proposed technique is described in section 3. Experimental results are discussed in section 4. Finally, section 5 draws the conclusions.

2. Related Work

Identification of noisy data is an important data preprocessing task for improving data quality. Many noise detection algorithms have been proposed for various applications [2, 5, 13, 16, 17, 21, 23, 24]. Among them, HCleaner [21], NOISERANK [16], Polishing method [17] and Error Detection and Impact-sensitive instance Ranking (EDIR) [23] are some well-known noisy value detection algorithms.

In HCleaner [21], data objects that are irrelevant or only weakly relevant are considered as noise. HCleaner is based on the concept of hyperclique patterns which consists of objects that are strongly similar to each other. The key idea behind this method is the use of hyperclique patterns as a filter to eliminate data objects that are not tightly connected to other data objects in the dataset.

The polishing method [17] is tolerant of some amount of noise in the data. Whereas filtering eliminates the noisy elements from the input, polishing corrects the noisy elements

rather than removing them. This method takes advantage of the interdependency between the components of a dataset to identify the noisy elements and suggest appropriate replacements. There are two stages in the polishing method: prediction stage and adjustment stage. In the prediction stage, elements in the data that are suspected of being noisy are identified together with their nominated replacement values. In the adjustment stage, this method selectively incorporates the nominated changes into the dataset.

Sluban *et al.* [16] propose an ensemble-based class noise detection method, NOISERANK, and a method for visual performance evaluation of class noise detection algorithms in the precision-recall space, named VIPER. NOISERANK is an expert-guided noise detection method. The user inspects the detected noisy instances and decides whether they are interesting outliers which lead to new insights in domain understanding, erroneous instances which should be removed, or instances with minor corrected errors to be reintroduced into the dataset. Four different classification filters are used in this method for detecting noisy instances in data. The classifiers are Naïve Bayes; Random Forest with 500 decision trees (RF500); Support Vector Machine; and Neural Network. For the Random Forest classifier, two variants of the High Agreement Random Forest noise detection algorithm (HARF-70 and HARF-80) perform the best. The classification filter is performed in a cross-validation manner, i.e. using repeatedly nine folds for training of a classifier and one complementary fold for class prediction where the incorrectly classified instances are considered to be noisy. NOISERANK also uses a saturation-based approach for noise filtering, called the saturation filter to determine the noisy instances. A saturation filter is constructed from two stages. The first stage is the saturation test. It first computes the complexity of the classification model for the given training set, and then it iteratively excludes one training example and computes the complexity of a classification model induced from the rest of the training examples. The examples which have the greatest effect on reducing the complexity of the classification model by their exclusions are labelled as the most noisy and are passed on to the second stage. The second stage, the noise filter, randomly chooses one from the noisiest examples and excludes it from the training set, while the other examples are returned to the example set. This is repeated as long as the saturation test finds noisy examples, meaning that a saturated subset has not yet been obtained. VIPER addresses the noise detection performance directly by measuring the precision, recall and the F-measure of different noise detection algorithms on data with known or injected noisy instances. It presents the visual performance evaluation in the precision-recall space.

Error Detection and Impact-sensitive instance Ranking (EDIR) [23] locates erroneous instances and attributes and rank suspicious instances based on their impact on system performance. At first, EDIR trains a benchmark classifier T from noisy dataset D. The instances that cannot be classified by T are treated as suspicious and forward to a subset S. Each instance contains n attributes $A_1, A_2, \ldots, A_n$ and each attribute A_i has V_i possible values. To rank instances in S, EDIR uses an impact measure based on Information-gain Ratio. Like polishing method, EDIR also changes the attribute value to correctly classify the record. However, EDIR differs from polishing method in several aspects. Unlike polishing method, EDIR can change two or three attributes values at a time if a suspicious record remains misclassified. The record is stored separately if it remains suspicious even after changing all combination of two or three values.

Bigdeli *et al.* [24] proposed the Collective Probabilistic Anomaly Detection (CPAD) method to increase the accuracy of the anomaly detection by focusing on arbitrary shape

clustering. In this method, the membership values of the new samples are calculated based on Gaussian Mixture model. The new samples are labelled as 'normal' or 'noise' based on the membership values. In CPAD method, the samples are clustered and labelled collectively as 'noise' or 'normal' instead of labelling them individually. The samples are clustered according to their group behaviours rather than individual characteristics. After that, the distance between two clusters are calculated using Gaussian Mixture model to label the clustered samples.

Co-appearance based Analysis for Incorrect Records and Attribute-values Detection (CAIRAD) [13] algorithm exploits the co-appearance between attributes values to detect noisy values of a dataset. To detect noisy values, the method generates a co-appearance matrix from the dataset and computes an expected co-appearance value for an attribute value. If the value from the co-appearance matrix and the expected co-appearance value are the same for an attribute value then this value is declared as a clean value, otherwise it is flagged as noisy. The RDCL method [2] uses kNN technique to classify a record. At first, dataset is divided into training and testing datasets. After that, RDCL classifies the record in testing dataset by using the majority class of its kNN records in the training dataset. RDCL identifies suspicious records whereas Polishing technique and EDIR detect both noisy attributes values and records.

In many practical scenarios, we can assume that most of the attributes of a dataset are clean and the volume of noise is low. Another observation is that noisy values have a random and independent nature and are not correlated to the occurrence of any other values of a dataset. It is also very rare that a noisy value will appear repeatedly as a result of the introduction of random noise.

3. Proposed Approach

In our approach, we first flag attribute values with occurrence frequency of 1 or 2 as potentially suspicious as it is unlikely that the same erroneous value occurs too frequently in a dataset. We then correct for simple typographical errors that occur in the flagged values using Levenshtein distance [8]. Specifically, if a flagged attribute value has a Levenshtein distance of 3 or less with a clean value (i.e. a flagged value) and it does not have the same Levenshtein distance with more than one clean value, then its value is changed to that of the clean value. The remaining flagged values are then considered as suspicious noisy values and are processed as follows.

Let the record having the suspicious noisy attribute value x for the attribute X be denoted by r. Depend on the suspicious noisy attribute value x, two subsets of records are created from the original dataset D. The first subset, denoted as S_s, contains a set of records from D where all attributes values are the same as record r except for the suspicious noisy attribute

value x. The second subset, denoted as S_d, contains a set of records from the remaining records in D where a varying number of attributes values are the same as record r. The noisiness of x is determined using records that are found in these two subsets.

To illustrate our method, Table 1 shows a sample traffic accident dataset (bold values of Table 1 are marked as suspicious noisy values). Subsets are built based on the suspicious noisy values ('Active' and 'Fog'). These subsets are shown in Tables 2, 3, 4 and 5.

Table 1. Sample traffic accident dataset

Record	Driver status	Weather condition	Accident address
R_1	Normal	Bad	Sanders
R_2	**Active**	Bad	Glendale
R_3	Normal	Stormy	Glendale
R_4	Abnormal	Stormy	Sanders
R_5	Drunk	Bad	Glendale
R_6	Normal	**Fog**	Sanders
R_7	Drunk	Bad	Glendale
R_8	Drunk	Bad	Glendale
R_9	Abnormal	Stormy	Sanders
R_{10}	Drunk	Bad	Glendale
R_{11}	Drunk	Bad	Glendale
R_{12}	Drunk	Bad	Glendale
R_{13}	Abnormal	Stormy	Sanders
R_{14}	Abnormal	Rainy	Sanders
R_{15}	Abnormal	Rainy	Sanders
R_{16}	Abnormal	Rainy	Sanders

Table 2. Subset S_s for attribute value 'Active' in 'Driver status'

Record	Driver status	Weather condition	Accident address
R_2	**Active**	Bad	Glendale
R_5	Drunk	Bad	Glendale
R_7	Drunk	Bad	Glendale

R_8	Drunk	Bad	Glendale
R_{10}	Drunk	Bad	Glendale
R_{11}	Drunk	Bad	Glendale
R_{12}	Drunk	Bad	Glendale

Table 3. Subset S_d for attribute value 'Active' in 'Driver status'

Record	Driver status	Weather condition	Accident address
R_1	Normal	Bad	Sanders
R_3	Normal	Stormy	Glendale

Table 4. Subset S_s for attribute value 'Fog' in 'Weather Condition'

Record	Driver status	Weather condition	Accident address
R_6	Normal	**Fog**	Sanders
R_1	Normal	Bad	Sanders

Table 5. SubsetS_d for attribute value 'Fog' in 'Weather Condition'

Record	Driver status	Weather condition	Accident address
R_3	Normal	Stormy	Glendale
R_4	Abnormal	Stormy	Sanders
R_9	Abnormal	Stormy	Sanders
R_{13}	Abnormal	Stormy	Sanders
R_{14}	Abnormal	Rainy	Sanders
R_{15}	Abnormal	Rainy	Sanders
R_{16}	Abnormal	Rainy	Sanders

The noisiness of 'Active' value is determined using *P-measure* and *S-measure* computed from its S_s and S_d. In our approach, the *P-measure* computes the probability of the suspicious noisy value being actually correct, and the *S-measure* computes the similarity between two values of an attribute. In the above example, if 'Active' in record R_2 is found to be noisy, the

correct value is either 'Drunk' or 'Normal'. Similarly, for the noisy 'Fog' value, its correct value is either 'Bad', 'Stormy' or 'Rainy'.

3.1. P-measure

To calculate the *P-measure*, we consider the two subsets S_s and S_d created based on the suspicious noisy attribute value x of record *r*.. Let P_s be the probability that an attribute value *x* remaining unchanged. Let P_{sp}be the probability that an attribute value *x* is changed into a different attribute value from the subsets S_s. Let P_{dp} be the probability that an attribute value *x* is changed into a different attribute value from the subsets S_d. Any attribute value x obviously satisfies the following relationship

$$P_s + P_{sp} + P_{dp} = 1 \qquad (1)$$

Next, we introduce two variables k_1 and k_2 for the noisy values. The variable k_1 specifies how many times an attribute value x is more likely to stay the same than to change to another value in S_s. The other parameter k_2 specifies how many times more likely x is to change to a value in S_s than the one in S_d. The two variables are defined as

$$k_1 = \frac{N_x}{N_{S_s} - N_x} \text{ , with } N_{S_s} > 1$$

$$k_2 = \frac{(N_{S_s} - N_x)(K-1)}{\sum_{i=1}^{N_{S_d}} \delta_i} \text{, with } N_{S_s} > 1 \text{ and } N_{S_d} > 0$$

where N_x is the frequency of attribute value *x* for the attribute *X* in S_s, N_{S_s} is the number of records in S_s, N_{S_d} is the number of records in S_d, *K* is the number of attributes in a record, and δ_i is the number of attributes values that are similar between record *r* and the i^{th} record of S_d. Hence, we can define the probabilities P_s, P_{sp}, P_{dp} using k_1 and k_2 as

$$P_s = k_1 \times P_{sp} = k_1 \times k_2 \times P_{dp} \qquad (2)$$

From the above, the probabilities of P_s, P_{sp}, and P_{dp} become

$$P_s = \frac{k_1 \times k_2}{k_1 \times k_2 + k_2 + 1} \qquad (3)$$

$$P_{sp} = \frac{k_2}{k_1 \times k_2 + k_2 + 1} \quad (4)$$

$$P_{dp} = \frac{1}{k_1 \times k_2 + k_2 + 1} \quad (5)$$

Substituting k_1 and k_2 in equations (4) and (5) we get,

$$P_{sp} = \frac{(N_{S_s} - N_x)(K-1)}{N_{S_s}(K-1) + \sum_{i=1}^{N_{S_d}} \delta_i} \quad (6)$$

$$P_{dp} = \frac{\sum_{i=1}^{N_{S_d}} \delta_i}{N_{S_s}(K-1) + \sum_{i=1}^{N_{S_d}} \delta_i} \quad (7)$$

Now, we can calculate the probability of an attribute value x changing to a value y_k in S_s or to a value z_k in S_d, where y_k are distinct value of attribute X except x in S_s and z_k are distinct value of attribute X in S_d. Let $P_{sp}(y_k)$ denotes the probability that an attribute value x is changed into y_k, then $\sum_{y_k} P_{sp}(y_k) = P_{sp}$. Since $N_{S_s} - N_x = \sum_{y_k} N_{y_k}$ where N_{y_k} is the frequency of attribute value y_k for the attribute X in S_s, we have

$$P_{sp}(y_k) = \frac{N_{y_k}(K-1)}{N_{S_s}(K-1) + \sum_{i=1}^{N_{S_d}} \delta_i} \quad (8)$$

Similarly, let $P_{dp}(z_k)$ denotes the probability that an attribute value x is changed into z_k. We have $\sum_{z_k} P_{dp}(z_k) = P_{dp}$. Let S_{z_k} denotes the subset of records containing value z_k of X in S_d, and N_{z_k} denotes the number of records in S_{z_k}. Let γ_j denotes the number of attributes values that are similar between record r and the j^{th} record of S_{z_k}. Then $P_{dp}(z_k)$ is given by

$$P_{dp}(z_k) = \frac{\sum_{j=1}^{N_{z_k}} \gamma_j}{N_{S_s}(K-1) + \sum_{i=1}^{N_{S_d}} \delta_i} \quad (9)$$

When the unchanged probability $P_s(x)$ of a suspicious attribute value x is higher than all of S_s attribute values probabilities ($P_{sp}(y_k)$, ……) and all of the S_d attribute values probabilities ($P_{dp}(z_k)$, ……), we declare this suspicious value x as a correct value.

To illustrate how the *P-measure* is computed, we compute the various probabilities for the suspicious noisy attribute value 'Fog' of record R_6 using Equations (3), (8), and (9), based on Tables 4 and 5. We have $P_s(Fog) = 0.182$, $P_{sp}(Bad) = 0.182, P_{dp}(Stormy) = 0.364$, $P_{dp}(Rainy) = 0.273$. As $P_s(Fog)$ is not higher than all the other probabilities, we cannot declare 'Fog' as a correct value for this record. Likewise, we cannot declare 'Active' in record R_2 as the correct value since $P_s(Active) = 0.125$, $P_{sp}(Drunk) = 0.750$, $P_{dp}(Normal) = 0.125$ from Tables 2 and 3.

Once *P-measure* is computed and the suspicious noisy attribute value cannot be declare as correct, we compute the *S-measure* of the suspicious noisy attribute value with the other possible values of the attribute in both S_s and S_d. The *S-measure* evaluates the similarity between two values of an attribute by taking into account their direct and transitive relationships. For example, we would compute the *S-measure* between 'Fog' and 'Bad', 'Fog' and 'Stormy', 'Fog' and 'Rainy'. The next section explains how the *S-measure* between two attribute values is computed.

3.2. S-measure

S-measure calculates the similarity between two attribute values of an attribute by looking at common neighbours (direct relationship, called 1st level similarity) of the two values and common neighbours of their neighbours (transitive relationship, called 2nd level similarity) [3]. The *S-measure* S_{ij} calculates the similarity between attribute values by creating a graph, where an edge exists between two vertices when the two corresponding values appear together in a record. Let $G = (V, E)$ be a graph created from the dataset consisting of records in both S_s and S_d, where the set of vertices V are given by the set of distinct attribute values in the dataset and an edge $e \in E$ *is* drawn between two vertices when both attribute values appear in a record. S_{ij} is the weighted sum of S'_{ij} (1st level similarity) and S''_{ij} (2nd level similarity) between two attribute values x_i and x_j of an attribute X

$$S_{ij} = C_1 \times S'_{ij} + C_2 \times S''_{ij} \qquad (10)$$

where C_1 and C_2 are the weights for the 1st level and 2nd level similarities, with $C_1 + C_2 = 1$. The 1st level similarity S'_{ij} is given by

$$S'_{ij} = \frac{\sum_{k=1}^{n} \sqrt{a_{ik} \times a_{kj}}}{\sqrt{d_i \times d_j}} \quad (11)$$

where a_{ik} = *l* if *l* edges occur between vertices *i* and *k*, and zero otherwise, *n* is the number of vertices in the graph *G*, and d_i is the degree of vertex i defined as the number of edges vertex *i* has. The 2nd level similarity S''_{ij} measures the transitive relationship for two attribute values of an attribute that are not directly connected to a common neighbour but are connected to pair of attribute values who have 1st level similarity S'_{ij} greater than a user defined threshold T. To calculate S''_{ij}, we first merge pairs of vertices with S'_{ij} greater than *T* in *G* to obtain a new graph *G'* and then re-apply equation (11) to *G'* (S'_{ij} in equation (11) now becomes the 2nd level similarity for *G'*).

To illustrate how *S-measure* is computed, we will use the records in Table 6 with vertex nodes labelled from 1 to 8 as an example. The 1st and 2nd levels *S-measure* graphs are constructed with respect to 'weather condition' attribute in Fig. 1 and Fig. 2.

Table 6. Aggregated table (Tables 4 and 5) for 'Fog' value with assigned vertex node number

Record	Driver status	Weather condition	Accident address
R_6	Normal (1)	**Fog** (3)	Sanders (7)
R_1	Normal (1)	Bad (4)	Sanders (7)
R_3	Normal (1)	Stormy (5)	Glendale (8)
R_4	Abnormal (2)	Stormy (5)	Sanders (7)
R_9	Abnormal (2)	Stormy (5)	Sanders (7)
R_{13}	Abnormal (2)	Stormy (5)	Sanders (7)
R_{14}	Abnormal (2)	Rainy (6)	Sanders (7)
R_{15}	Abnormal (2)	Rainy (6)	Sanders (7)
R_{16}	Abnormal (2)	Rainy (6)	Sanders (7)

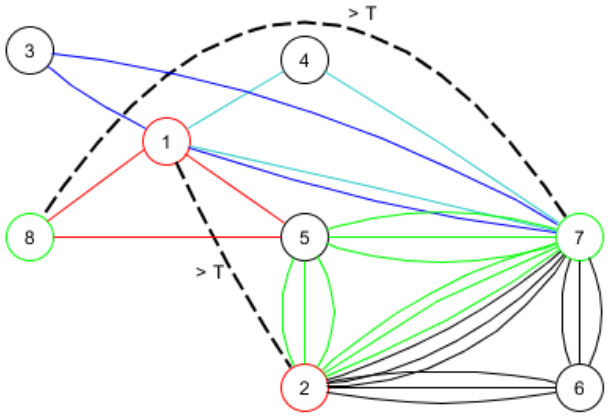


Figure 1: 1st level *S-measure* graph with respect to 'weather condition' attribute constructed from Table 6

From the graph in Figure 1, we can see that nodes 3 and 4 have nodes 1, and 7 as common neighbours. Hence the 1st level similarity between nodes 3 and 4 is $S'_{3,4} = \frac{\sqrt{(1\times1)}+\sqrt{(1\times1)}}{\sqrt{2\times2}} = 1$. Similarly, we can calculate the 1st level similarities for all node pairs within the graph as: $S'_{3,5} = 0.68$ (with neighbour nodes 1, 7), $S'_{3,6} = 0.50$ (with neighbour node 7). Let T=0.45, C_1=0.65, C_2=0.35 (see section 4.3 for this choice). In Fig. 1, we connect nodes having 1st level similarities greater than the threshold T by dotted lines.

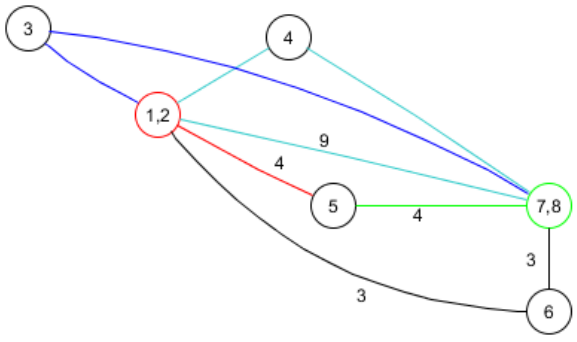


Figure 2: 2nd level *S-measure* graph with respect to 'weather condition' attribute constructed from Figure 1

To calculate the 2nd level similarity S''_{ij} between nodes i and j, we find all pairs of nodes (k, l) common to nodes i and j and with $S'_{kl} > T$, and merge each pair of nodes into a single vertex as shown in Figure 2. Here, vertices 1 and 2 are merged because $S'_{1,2} > T$ and vertices 7 and 8 are also merged. In Fig. 2, the number of multiple edges between two vertices is

denoted by a number associated with the edge for ease of visualization. For nodes 3 and 6, using Equation (11) (but with S'_{ij} replaced by S''_{ij}) we have $S''_{3,6} = \frac{\sqrt{(3\times1)}+\sqrt{(3\times1)}}{\sqrt{2\times6}} = 1$ (with neighbour nodes 1, 2, 7, and 8), $S''_{3,5} = 1$ (with neighbour nodes 1, 2, 7, and 8), and $S''_{3,4} = 1$. Likewise, we can calculate 2nd level similarity for all node pairs. In this example, the *S-measure* between (Fog and Bad), (Fog and Stormy) and (Fog and Rainy) are 1, 0.79 and 0.68 respectively.

Once the most similar attribute value, say *y*, is found using *S-measure*, we check the *P-measure* of the suspicious noisy attribute value *x* changing into *y*. If $P_{sp}(y)$ or $P_{dp}(y)$ is higher than the unchanged probability $P_s(x)$ of *x*, the value x is changed to *y*, otherwise we declare x as a correct value. Specifically, let $Y = \{y_1, y_2, \ldots, y_k\}$ be the set of all *k* possible attribute values of the suspicious value *x*, and let $y = max_{y \in Y} S_{ij}(x, Y)$, then

$$x = \begin{cases} y & \text{if } (P_{sp}(y) \text{ or } P_{dp}(y)) > P_s(x) \\ x & \text{otherwise} \end{cases} \qquad (12)$$

By considering both the *S-measure* and the *P-measure*, the suspicious value 'Fog' in record R_6 is declared to be the correct value, whereas the suspicious value 'Active' in R_2 is replaced by 'Drunk'.

3.3. Proposed Algorithm

The proposed algorithm is summarized below.

NoiseCleaner Algorithm:

(1) Count the frequency of each distinct attribute value in dataset *D* and flag values whose frequencies are smaller than 3 as suspicious.

(2) Correct for simple typographical errors that occur in the marked values using Levenshtein distance.

(3) For each suspicious attribute value *x* in *D*

(3.1) Create the corresponding set of records S_s and S_d

(3.2) Calculate the following probabilities from records in S_s and S_d:

(3.2.1) Calculate $P_s(x)$ using Equation (3)

(3.2.2) For each distinct value y_k of *X* from S_s, calculate $P_{sp}(y_k)$ using Equation (8)

(3.2.3) For each distinct value z_k of X from S_d, calculate $P_{dp}(z_k)$ using Equation (9)

(3.3) Declare x as correct if $P_s(x)$ is greater than all of $P_{sp}(y_k)$ and $P_{dp}(z_k)$ and break;

Else

(3.3.1) Calculate $S_{ij}(x, y_k)$ and $S_{ij}(x, z_k)$ using Equation (10)

(3.3.2) Select $y = max_{y \in Y} S_{ij}(x, Y)$ where $Y = \{y_k, z_k\}$

(3.3.3) Update x using Equation (12)

4. Experimental results and discussion

4.1. Datasets

We performed experiment on four datasets. Two road crash datasets are taken from the State of Queensland, Australia [19], and other two datasets are taken from the motor vehicle crashes of the New York State, United States [20]. In the four datasets, most of the attributes are categorical. We listed the four datasets in Table 7.

Table 7. Description of datasets

Abbreviation	Dataset name	#Records	#Categorical attributes	#Numerical attributes	As on date
RCL	Road Crash Locations [19]	251705	30	20	31 July, 2015
RC	Road Casualties [19]	15744	6	1	31 July, 2015
MCI	Motor Vehicle Crash-Case Information: 2011 [20]	13889	17	1	24 September, 2014
MII	Motor Vehicle Crash-Individual Information: 2011 [20]	17858	10	3	24September, 2014

4.2. Performance measures

Several performance measures are commonly used to evaluate the performance of noisy value detection algorithms [13, 16]. A popular measure is *precision*. *Precision* is the ratio of the number of relevant records retrieved to the total number of irrelevant and relevant records retrieved.

$$Precision = \frac{number\ of\ true\ noisy\ instances\ detected}{number\ of\ all\ instances\ identified\ as\ noisy} \quad (13)$$

Another useful measure is *recall*. *Recall* is the ratio of the number of relevant records retrieved to the total number of relevant records in the database.

$$Recall = \frac{number\ of\ true\ noisy\ instances\ detected}{number\ of\ all\ noisy\ instances\ in\ the\ dataset} \quad (14)$$

Precision and *Recall* are inversely related. *F-measure* is the weighted harmonic mean of *precision* and *recall* and is defined as in Equation (15).

$$F = 2 \times \frac{(precision \times recall)}{(precision + recall)} \quad (15)$$

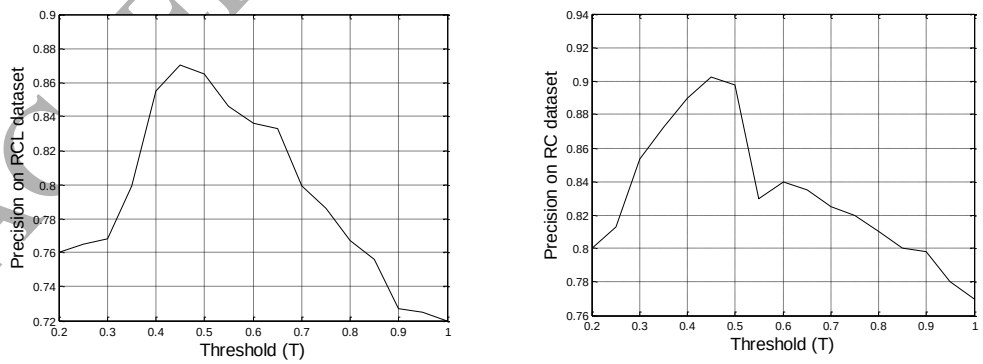
Equation (15) is also known as the F_1 measure or traditional F-measure or balanced F-score, because *recall* and *precision* are weighted evenly. It is possible to give emphasis to precision or recall by using

$$F_\beta = (1 + \beta^2) \frac{(precision \times recall)}{((\beta^2 \times precision) + recall)} \quad (16)$$

By setting the parameter β ($\beta > 0$) the user can assign more importance to either *precision* or *recall* in the computation of the *F-measure*. Two others commonly used F measures are the F_2 measure, which weights *recall* higher than *precision*, and the $F_{0.5}$ measure, which puts more emphasis on *precision* than *recall*.

4.3. Parameter selection for *S-measure*

S-measure requires the setting of three parameters: T, C_1, and C_2. Using the precision measure, we analyse the four datasets to select the best threshold parameter T. The result is shown in Figure 3 and we set T=0.45. In Table 8, we see that the optimum values for C_1, and C_2 are 0.65 and 0.35, respectively. It can be seen that the parameter values are not highly dependent on the datasets, and these values can be used as default for many different datasets.



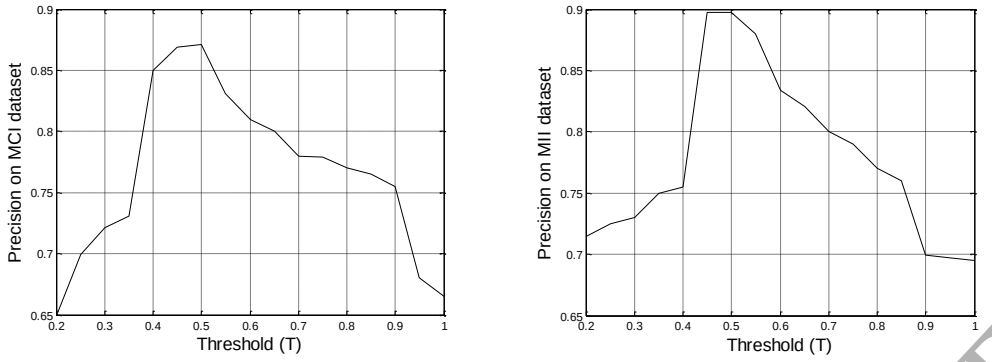


Figure 3: Threshold parameter T on four datasets

Table 8: C_1 and C_2parameters selection using *precision*

Parameters		Datasets			
C_1	C_2	*RCL*	*RC*	*MCI*	*MII*
1.00	0.00	0.7032	0.7006	0.6995	0.7905
0.95	0.05	0.7035	0.7005	0.7001	0.7924
0.90	0.10	0.7050	0.7152	0.7010	0.7990
0.85	0.15	0.8060	0.7835	0.7234	0.8012
0.80	0.20	0.8065	0.7989	0.7346	0.8044
0.75	0.25	0.8275	0.8455	0.8412	0.8342
0.70	**0.30**	**0.8500**	**0.8511**	**0.8502**	**0.8512**
0.65	**0.35**	**0.8712**	**0.9099**	**0.8741**	**0.9095**
0.60	**0.40**	**0.8401**	**0.8444**	**0.8788**	**0.8441**
0.55	0.45	0.8038	0.8331	0.8742	0.8211
0.50	0.50	0.7681	0.8112	0.8578	0.8000
0.45	0.55	0.7610	0.7908	0.8022	0.7989
0.40	0.60	0.6211	0.7511	0.7010	0.7554
0.35	0.65	0.5904	0.6904	0.6520	0.6987
0.30	0.70	0.4801	0.5632	0.6011	0.6564
0.25	0.75	0.4812	0.5323	0.5902	0.6226
0.20	0.80	0.4913	0.5312	0.5822	0.5012
0.15	0.85	0.4012	0.5011	0.5012	0.4812
0.10	0.90	0.4109	0.4978	0.4824	0.4788
0.05	0.95	0.4147	0.4876	0.6204	0.4546
0.00	1.00	0.4098	0.4524	0.6202	0.4004

4.4. Noisy values simulation

We use four types of noisy patterns in our test datasets: simple, medium, complex, and blended [3, 5]. In a simple pattern, a record can have at most one noisy value, whereas in a medium pattern, a record can have noisy values for up to 50% of the attributes. Similarly, in a complex pattern, a record can have noisy values for up to 80% of the attributes. A blended pattern contains mixture of three patterns (simple pattern 25%, 50% with medium pattern, and 25% with complex pattern). For each of the noisy pattern, we use four different noisy

ratios (2%, 4%, 6% and 8%) where x% noisy ratio means x% of the attribute values of a dataset are noisy. We use two types of noisy models, namely overall and uniformly distributed (UD). In the UD noisy model, each attribute has equal number of noisy attribute values. However, in the overall model, noisy attribute values are not equally distributed among the attributes, and in the worst case all noisy attribute values can belong to a single attribute. Additionally, for each test dataset, 1% of the attribute values of a dataset are created with typographical errors, where errors of 1, 2 or 3 characters are randomly introduced.

In our experiments, we create 32 noisy combinations (4 noisy ratios × 4 noisy patterns × 2 noisy models). For each combination, we generate five datasets i.e. in total we create 160 noisy datasets (32 combinations × 5 datasets per combination) for each real dataset as shown in Table 9.

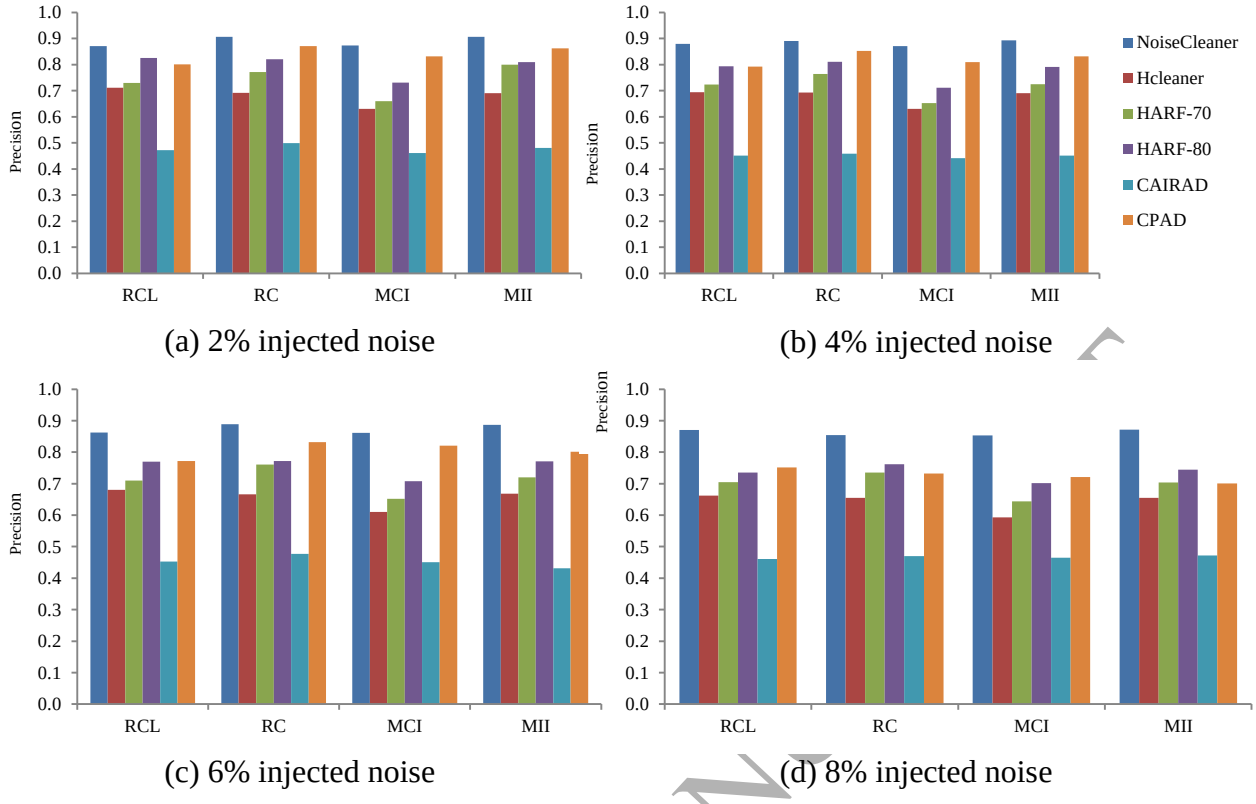
Table 9. Noisy value simulation

Noisy patterns	Number of attributes having noisy values in a record	Typographical error	Noise ratios	Noisy model	Number of datasets for each pattern
Simple	1	1% with randomly 1, 2, or 3 characters	2%, 4%, 8% and 10%	Overall and Uniformly distributed	5
Medium	Up to 50%				
Complex	Up to 80%				
Blended	Simple-25%, Medium-50% and Complex-25%				

4.5. Experimental results

We compare our proposed algorithm NoiseCleaner with five noisy values identification methods, namely HARF-80 and HARF-70 from NOISERANK [16], HCleaner [21], CPAD [24], and CAIRAD [13]. In NOISERANK, there are several ensemble methods. Among them, HARF-80 and HARF-70 performed the best compared to the other methods and we take these two methods for comparison with our NoiseCleaner algorithm. Simple typographical errors are corrected using Levenshtein distance as a pre-processing for all methods.

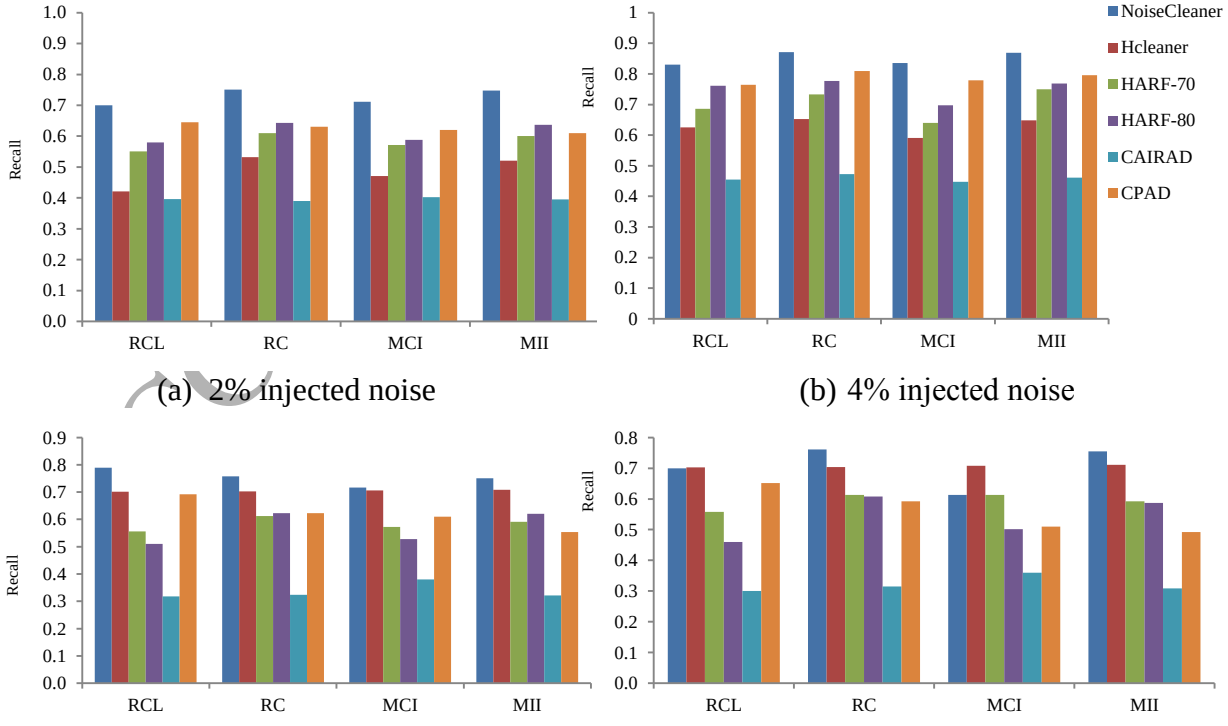
We present the noisy values detection accuracy (*Precision*) of NoiseCleaner, HCleaner, HARF-70, HARF-80, CPAD, and CAIRAD in Fig. 4. For each dataset, we show 4 combinations of four noisy ratios (2%, 4%, 6%, and 8%), and one noisy pattern (medium). Fig. 5 shows the corresponding *recall* result and Fig. 6 shows the $F_{0.5}$ result. From these results, it can be seen that NoiseCleaner performs significantly better than HARF-80, HARF-70, HCleaner, CPAD, and CAIRAD on all 4 datasets.



(a) 2% injected noise (b) 4% injected noise

(c) 6% injected noise (d) 8% injected noise

Figure 4: The *precision* results of different noisy value detection algorithms on 4 datasets for various 'noisy ratios' and 'medium noisy pattern' with confidence level 95 percent.



(a) 2% injected noise (b) 4% injected noise

(c) 6% injected noise (d) 8% injected noise

Figure 5: The *recall* results of different noisy value detection algorithms on 4 datasets for

various ‘noisy ratios’ and ‘medium noisy pattern’ with confidence level 95 percent.

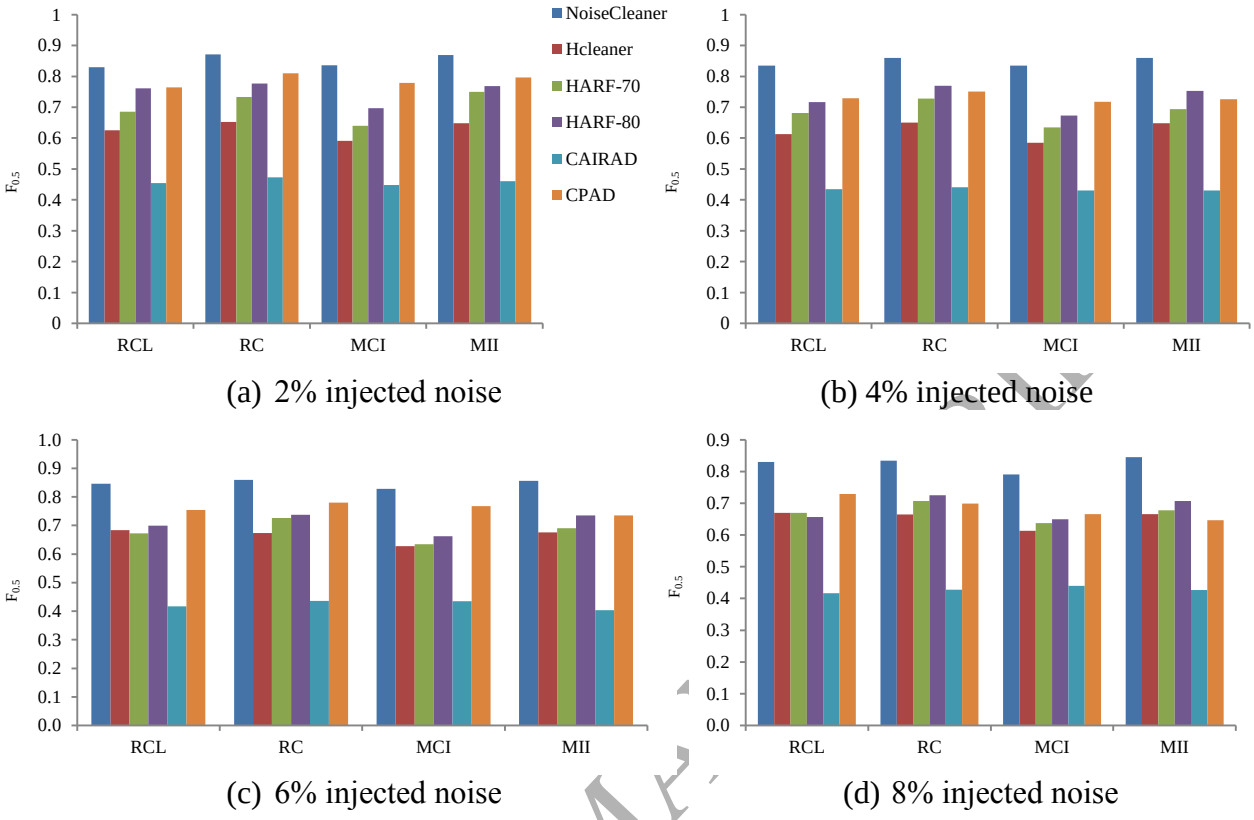


Figure 6: The $F_{0.5}$ results of different noisy value detection algorithms on 4 datasets for various ‘noisy ratios’ and ‘medium noisy pattern’ with confidence level 95 percent.

The noisy value detection accuracy of NoiseCleaner, HARF-70, HARF-80, CPAD, and HCleaner on the 4 datasets in terms of *precision* for the 32 noisy combinations is presented in Tables 10-13. Each value in these tables is the average result over 5 datasets generated for each combination of noisy ratio, noisy model, and noisy pattern. In these tables, bold values mark the best result among all methods and italic values represent the second best result. It can be seen that NoiseCleaner performs significantly better than HARF-80, HARF-70, CPAD, and HCleaner, by wining in all cases. HARF-80 is the next best performing algorithm, losing to HARF-70 in only 4 cases on the MII dataset (Table 11).

Table 10: Performance on RC dataset

Noise combination			Precision					
			NoiseCleaner	HARF-80	HARF-70	HCleaner	CAIRAD	CPAD
2%	Overall	Simple	**0.9087±0.01**	*0.8233±0.02*	0.7732±0.02	0.6896±0.01	0.4980±0.02	*0.8730±0.01*
		Medium	**0.9084±0.02**	*0.8246±0.01*	0.7752±0.03	0.6891±0.02	0.4980±0.02	*0.8712±0.02*
		Complex	**0.9082±0.01**	*0.8144±0.04*	0.7688±0.01	0.6895±0.01	0.4957±0.02	*0.8704±0.03*
		Blended	**0.9078±0.03**	*0.8135±0.05*	0.7678±0.02	0.6891±0.04	0.4951±0.03	*0.8700±0.10*

	UD	Simple	**0.9065±0.01**	*0.8237±0.03*	0.7776±0.02	0.6898±0.01	0.4901±0.02	*0.8611±0.02*
		Medium	**0.9062±0.04**	*0.8245±0.01*	0.7728±0.04	0.6865±0.04	0.4900±0.02	*0.8601±0.04*
		Complex	**0.9059±0.02**	*0.8189±0.05*	0.7701±0.03	0.6894±0.05	0.4886±0.03	*0.8523±0.03*
		Blended	**0.9047±0.01**	*0.8178±0.02*	0.7709±0.05	0.6898±0.01	0.4880±0.04	*0.8520±0.01*
4%	Overall	Simple	**0.8901±0.01**	*0.8128±0.02*	0.7662±0.01	0.6918±0.01	0.4845±0.02	*0.8525±0.03*
		Medium	**0.8910±0.03**	*0.8119±0.03*	0.7669±0.01	0.6904±0.03	0.4843±0.01	*0.8521±0.05*
		Complex	**0.8912±0.02**	*0.8127±0.04*	0.7661±0.03	0.6808±0.04	0.4842±0.03	*0.8520±0.03*
		Blended	**0.8903±0.01**	*0.8110±0.03*	0.7664±0.01	0.6802±0.02	0.4840±0.03	*0.8419±0.01*
	UD	Simple	**0.9001±0.02**	*0.8104±0.01*	0.7663±0.02	0.6887±0.01	0.4745±0.03	*0.8400±0.03*
		Medium	**0.8900±0.04**	*0.8061±0.02*	0.7648±0.01	0.6745±0.06	0.4739±0.02	*0.8335±0.02*
		Complex	**0.8923±0.03**	*0.8098±0.05*	0.7646±0.05	0.6732±0.02	0.4730±0.01	*0.8330±0.02*
		Blended	**0.8921±0.01**	*0.8040±0.06*	0.7642±0.01	0.6731±0.03	0.4704±0.02	*0.8313±0.01*
6%	Overall	Simple	**0.8910±0.01**	*0.8032±0.02*	0.7640±0.03	0.6702±0.02	0.4730±0.02	*0.8303±0.04*
		Medium	**0.8909±0.02**	*0.8031±0.02*	0.7631±0.01	0.6688±0.04	0.4729±0.01	*0.8300±0.03*
		Complex	**0.8806±0.06**	*0.7692±0.05*	0.7638±0.04	0.6650±0.02	0.4720±0.02	*0.8289±0.01*
		Blended	**0.8801±0.02**	*0.7693±0.04*	0.7620±0.01	0.6615±0.04	0.4720±0.03	*0.8276±0.06*
	UD	Simple	**0.8802±0.01**	*0.7793±0.03*	0.7620±0.04	0.6608±0.02	0.4718±0.04	*0.8270±0.06*
		Medium	**0.8801±0.07**	*0.7692±0.02*	0.7578±0.02	0.6602±0.06	0.4710±0.01	*0.8219±0.03*
		Complex	**0.8800±0.04**	*0.7691±0.01*	0.7540±0.08	0.6604±0.05	0.4709±0.02	*0.8209±0.02*
		Blended	**0.8800±0.02**	*0.7580±0.06*	0.7400±0.10	0.6568±0.01	0.4708±0.03	*0.8203±0.02*
8%	Overall	Simple	**0.8901±0.02**	*0.7621±0.03*	0.7400±0.01	0.6672±0.03	0.4710±0.02	*0.7424±0.02*
		Medium	**0.8886±0.04**	*0.7605±0.04*	0.7379±0.03	0.6648±0.02	0.4708±0.03	*0.7412±0.01*
		Complex	**0.8885±0.01**	*0.7590±0.06*	0.7358±0.05	0.6535±0.06	0.4707±0.04	*0.7402±0.02*
		Blended	**0.8880±0.05**	*0.7538±0.07*	0.7356±0.06	0.6522±0.07	0.4706±0.01	*0.7400±0.04*
	UD	Simple	**0.8880±0.03**	*0.7910±0.02*	0.7301±0.07	0.6610±0.07	0.4705±0.02	*0.7345±0.01*
		Medium	**0.8879±0.05**	*0.7903±0.05*	0.7280±0.04	0.6504±0.03	0.4700±0.03	*0.7340±0.03*
		Complex	**0.8877±0.03**	*0.7940±0.03*	0.7020±0.04	0.6501±0.04	0.4700±0.04	*0.7329±0.02*
		Blended	**0.8872±0.06**	*0.7945±0.07*	0.7210±0.05	0.6500±0.02	0.4880±0.02	*0.7308±0.02*

Table 11: Performance on MII dataset

			Precision					
Noise combination			NoiseCleaner	HARF-80	HARF-70	HCleaner	CAIRAD	CPAD
2%	Overall	Simple	**0.9060±0.01**	0.8108±0.02	*0.8120±0.01*	0.6935±0.02	0.4896±0.02	*0.8642±0.01*
		Medium	**0.9058±0.03**	0.8103±0.01	*0.8030±0.03*	0.7030±0.04	0.4894±0.01	*0.8630±0.02*
		Complex	**0.9057±0.04**	*0.8111±0.03*	0.8015±0.03	0.7036±0.05	0.4892±0.04	*0.8625±0.03*
		Blended	**0.9055±0.06**	*0.8110±0.02*	0.8004±0.02	0.7040±0.02	0.4890±0.03	*0.8620±0.03*
	UD	Simple	**0.9050±0.02**	0.8014±0.04	*0.8019±0.04*	0.7114±0.05	0.4889±0.04	*0.8615±0.02*
		Medium	**0.9048±0.01**	0.8024±0.05	*0.8091±0.05*	0.7018±0.10	0.4888±0.08	*0.8612±0.01*
		Complex	**0.9047±0.03**	*0.7929±0.06*	0.7845±0.06	0.6922±0.05	0.4888±0.03	*0.8610±0.02*
		Blended	**0.9044±0.04**	*0.7930±0.07*	0.7231±0.03	0.6910±0.04	0.4887±0.01	*0.8601±0.01*
4%	Overall	Simple	**0.9046±0.04**	*0.7993±0.02*	0.7301±0.02	0.6901±0.03	0.4890±0.01	*0.8401±0.02*
		Medium	**0.8943±0.03**	*0.7981±0.03*	0.7302±0.03	0.6887±0.02	0.4891±0.02	*0.8400±0.01*
		Complex	**0.8942±0.02**	*0.7850±0.05*	0.7240±0.04	0.6780±0.04	0.4888±0.03	*0.8398±0.03*
		Blended	**0.8940±0.04**	*0.7895±0.01*	0.7148±0.05	0.6781±0.06	0.4883±0.02	*0.8390±0.05*

	UD	Simple	**0.8935±0.05**	*0.7896±0.06*	0.7223±0.04	0.6878±0.02	0.4884±0.01	*0.8356±0.06*
		Medium	**0.8934±0.06**	*0.7888±0.04*	0.7215±0.03	0.6858±0.07	0.4878±0.02	*0.8350±0.02*
		Complex	**0.8933±0.07**	*0.7864±0.01*	0.7122±0.02	0.6796±0.09	0.4870±0.04	*0.8346±0.01*
		Blended	**0.8931±0.08**	*0.7893±0.05*	0.7108±0.06	0.6743±0.10	0.4871±0.02	*0.8340±0.03*
6%	Overall	Simple	**0.8924±0.02**	*0.7790±0.02*	0.7202±0.06	0.6810±0.04	0.4873±0.03	*0.8220±0.01*
		Medium	**0.8922±0.01**	*0.7758±0.03*	0.7238±0.11	0.6800±0.03	0.4871±0.02	*0.8212±0.02*
		Complex	**0.8921±0.03**	*0.7740±0.05*	0.7126±0.14	0.6698±0.02	0.4870±0.03	*0.8104±0.04*
		Blended	**0.8920±0.05**	*0.7690±0.04*	0.7082±0.05	0.6690±0.02	0.4869±0.02	*0.8101±0.02*
	UD	Simple	**0.8923±0.04**	*0.7680±0.06*	0.7100±0.07	0.6786±0.08	0.4868±0.03	*0.8100±0.01*
		Medium	**0.8922±0.01**	*0.7588±0.03*	0.7098±0.08	0.6782±0.05	0.4864±0.04	*0.8076±0.02*
		Complex	**0.8922±0.02**	*0.7560±0.06*	0.7090±0.01	0.6688±0.03	0.4863±0.01	*0.8074±0.05*
		Blended	**0.8920±0.01**	*0.7520±0.10*	0.7084±0.03	0.6690±0.06	0.4862±0.01	*0.8070±0.01*
8%	Overall	Simple	**0.8819±0.03**	*0.7577±0.01*	0.7190±0.02	0.6610±0.03	0.4866±0.04	*0.7132±0.02*
		Medium	**0.8712±0.02**	*0.7590±0.08*	0.7130±0.05	0.6603±0.06	0.4864±0.02	*0.7124±0.01*
		Complex	**0.8710±0.05**	*0.7441±0.04*	0.7014±0.06	0.6568±0.10	0.4864±0.03	*0.7120±0.02*
		Blended	**0.8708±0.02**	*0.7380±0.05*	0.7010±0.07	0.6545±0.11	0.4860±0.02	*0.7103±0.03*
	UD	Simple	**0.8713±0.04**	*0.7461±0.02*	0.7004±0.06	0.6600±0.12	0.4861±0.03	*0.7100±0.02*
		Medium	**0.8709±0.06**	*0.7456±0.06*	0.6989±0.03	0.6555±0.02	0.4859±0.05	*0.7045±0.01*
		Complex	**0.8703±0.03**	*0.7438±0.02*	0.6910±0.02	0.6515±0.06	0.4850±0.02	*0.7040±0.03*
		Blended	**0.8732±0.02**	*0.7390±0.05*	0.6900±0.03	0.6508±0.04	0.4840±0.02	*0.7030±0.03*

Table 12: Performance on RCL dataset

			Precision					
Noise combination			NoiseCleaner	HARF-80	HARF-70	HCleaner	CAIRAD	CPAD
2%	Overall	Simple	**0.8701±0.02**	*0.8212±0.03*	0.7360±0.03	0.7240±0.04	0.4732±0.03	*0.8011±0.02*
		Medium	**0.8702±0.02**	*0.8210±0.04*	0.7320±0.03	0.7184±0.04	0.4730±0.03	*0.8010±0.01*
		Complex	**0.8700±0.06**	*0.8090±0.04*	0.7321±0.13	0.7187±0.08	0.4729±0.02	*0.8009±0.02*
		Blended	**0.8700±0.05**	*0.8060±0.02*	0.7325±0.11	0.7185±0.11	0.4728±0.08	*0.8004±0.04*
	UD	Simple	**0.8702±0.02**	*0.8221±0.03*	0.7311±0.04	0.7138±0.06	0.4727±0.05	*0.8003±0.01*
		Medium	**0.8790±0.04**	*0.8145±0.04*	0.7216±0.04	0.7099±0.05	0.4726±0.03	*0.8002±0.03*
		Complex	**0.8782±0.10**	*0.8126±0.01*	0.7241±0.11	0.7085±0.02	0.4724±0.01	*0.8001±0.02*
		Blended	**0.8680±0.09**	*0.8121±0.02*	0.7138±0.10	0.7080±0.01	0.4725±0.02	*0.8000±0.02*
4%	Overall	Simple	**0.8700±0.03**	*0.8081±0.06*	0.7240±0.05	0.6966±0.04	0.4722±0.05	*0.7948±0.03*
		Medium	**0.8689±0.02**	*0.8085±0.04*	0.7245±0.06	0.6950±0.03	0.4721±0.02	*0.7945±0.04*
		Complex	**0.8688±0.04**	*0.7989±0.03*	0.7205±0.10	0.6920±0.04	0.4722±0.01	*0.7942±0.02*
		Blended	**0.8683±0.06**	*0.7988±0.02*	0.7165±0.11	0.6924±0.06	0.4719±0.03	*0.7934±0.08*
	UD	Simple	**0.8688±0.03**	*0.7937±0.01*	0.7182±0.07	0.6960±0.07	0.4710±0.04	*0.7914±0.04*
		Medium	**0.8676±0.04**	*0.7896±0.08*	0.7070±0.06	0.6887±0.06	0.4709±0.02	*0.7911±0.03*
		Complex	**0.8673±0.02**	*0.7888±0.01*	0.7055±0.17	0.6813±0.06	0.4707±0.02	*0.7909±0.02*
		Blended	**0.8670±0.01**	*0.7890±0.02*	0.7017±0.11	0.6810±0.04	0.4706±0.01	*0.7903±0.03*
6%	Overall	Simple	**0.8682±0.05**	*0.7790±0.01*	0.7108±0.06	0.6834±0.03	0.4712±0.02	*0.7835±0.02*
		Medium	**0.8681±0.04**	*0.7783±0.04*	0.7141±0.08	0.6809±0.10	0.4710±0.01	*0.7831±0.04*
		Complex	**0.8679±0.08**	*0.7758±0.03*	0.7036±0.16	0.6800±0.06	0.4709±0.02	*0.7823±0.03*
		Blended	**0.8678±0.09**	*0.7650±0.02*	0.7001±0.15	0.6740±0.07	0.4708±0.05	*0.7821±0.03*

	UD	Simple	**0.8669±0.04**	*0.7640±0.03*	0.7104±0.06	0.6806±0.10	0.4707±0.06	*0.7787±0.02*
		Medium	**0.8667±0.02**	*0.7620±0.02*	0.7101±0.07	0.6790±0.11	0.4705±0.03	*0.7734±0.01*
		Complex	**0.8663±0.03**	*0.7409±0.04*	0.7080±0.14	0.6708±0.12	0.4700±0.03	*0.7713±0.02*
		Blended	**0.8661±0.06**	*0.7401±0.08*	0.6998±0.12	0.6701±0.04	0.4701±0.02	*0.7700±0.02*
8%	Overall	Simple	**0.8660±0.01**	*0.7550±0.06*	0.7066±0.06	0.6723±0.04	0.4707±0.02	*0.7621±0.01*
		Medium	**0.8559±0.04**	*0.7402±0.04*	0.7023±0.04	0.6661±0.11	0.4706±0.02	*0.7511±0.03*
		Complex	**0.8553±0.02**	*0.7401±0.06*	0.7015±0.05	0.6637±0.05	0.4706±0.01	*0.7503±0.02*
		Blended	**0.8551±0.07**	*0.7400±0.07*	0.6987±0.12	0.6620±0.04	0.4705±0.03	*0.7501±0.04*
	UD	Simple	**0.8643±0.02**	*0.7435±0.07*	0.7046±0.13	0.6623±0.04	0.4704±0.02	*0.7500±0.03*
		Medium	**0.8631±0.04**	*0.7395±0.03*	0.7032±0.11	0.6610±0.06	0.4703±0.04	*0.7498±0.02*
		Complex	**0.8524±0.02**	*0.7107±0.01*	0.7004±0.08	0.6585±0.03	0.4700±0.05	*0.7490±0.02*
		Blended	**0.8521±0.03**	*0.7100±0.06*	0.6900±0.18	0.6502±0.10	0.4700±0.01	*0.7409±0.02*

Table 13: Performance on MCI dataset

Noise combination			Precision					
			NoiseCleaner	HARF-80	HARF-70	HCleaner	CAIRAD	CPAD
2%	Overall	Simple	**0.8731±0.01**	*0.7367±0.01*	0.6753±0.03	0.6305±0.06	0.4953±0.04	*0.8352±0.03*
		Medium	**0.8728±0.05**	*0.7361±0.04*	0.6630±0.12	0.6304±0.12	0.4952±0.04	*0.8309±0.02*
		Complex	**0.8725±0.02**	*0.7354±0.02*	0.6617±0.10	0.6301±0.18	0.4950±0.01	*0.8302±0.01*
		Blended	**0.8716±0.03**	*0.7252±0.01*	0.6587±0.12	0.6300±0.03	0.4949±0.02	*0.8301±0.02*
	UD	Simple	**0.8720±0.02**	*0.7248±0.04*	0.6710±0.13	0.6302±0.16	0.4939±0.03	*0.8300±0.02*
		Medium	**0.8721±0.04**	*0.7263±0.05*	0.6508±0.09	0.6271±0.09	0.4938±0.02	*0.8287±0.01*
		Complex	**0.8715±0.05**	*0.7182±0.03*	0.6500±0.08	0.6108±0.06	0.4836±0.01	*0.8267±0.02*
		Blended	**0.8712±0.01**	*0.7101±0.01*	0.6489±0.12	0.6101±0.08	0.4835±0.02	*0.8265±0.03*
4%	Overall	Simple	**0.8707±0.04**	*0.7285±0.07*	0.6587±0.10	0.6257±0.11	0.4835±0.01	*0.8262±0.02*
		Medium	**0.8706±0.02**	*0.7281±0.02*	0.6589±0.06	0.6338±0.07	0.4833±0.02	*0.8243±0.01*
		Complex	**0.8602±0.03**	*0.7187±0.03*	0.6567±0.06	0.6158±0.04	0.4830±0.02	*0.8240±0.02*
		Blended	**0.8600±0.04**	*0.7068±0.04*	0.6561±0.05	0.6178±0.05	0.4831±0.02	*0.8224±0.03*
	UD	Simple	**0.8725±0.03**	*0.7167±0.02*	0.6583±0.04	0.6280±0.06	0.4726±0.05	*0.8206±0.02*
		Medium	**0.8628±0.06**	*0.7160±0.03*	0.6577±0.02	0.6228±0.11	0.4724±0.03	*0.8201±0.02*
		Complex	**0.8625±0.04**	*0.7052±0.03*	0.6460±0.10	0.6191±0.10	0.4723±0.02	*0.8200±0.01*
		Blended	**0.8614±0.03**	*0.7000±0.01*	0.6450±0.01	0.6187±0.02	0.4720±0.01	*0.8200±0.02*
6%	Overall	Simple	**0.8722±0.08**	*0.7172±0.03*	0.6649±0.03	0.6303±0.05	0.4718±0.03	*0.8187±0.01*
		Medium	**0.8720±0.04**	*0.7171±0.04*	0.6608±0.11	0.6312±0.01	0.4719±0.02	*0.8176±0.03*
		Complex	**0.8618±0.03**	*0.7055±0.05*	0.6600±0.04	0.6176±0.02	0.4711±0.01	*0.8164±0.02*
		Blended	**0.8616±0.04**	*0.6943±0.06*	0.6588±0.01	0.6074±0.03	0.4711±0.04	*0.8160±0.01*
	UD	Simple	**0.8712±0.05**	*0.7100±0.03*	0.6610±0.06	0.6395±0.05	0.4710±0.02	*0.8043±0.01*
		Medium	**0.8600±0.06**	*0.7087±0.03*	0.6584±0.06	0.6313±0.06	0.4709±0.02	*0.8021±0.02*
		Complex	**0.8565±0.07**	*0.7047±0.02*	0.6556±0.04	0.6041±0.02	0.4703±0.03	*0.8001±0.01*
		Blended	**0.8580±0.08**	*0.6998±0.04*	0.6497±0.02	0.6000±0.03	0.4702±0.01	*0.8000±0.04*
8%	Overall	Simple	**0.8600±0.02**	*0.7141±0.04*	0.6581±0.07	0.6201±0.11	0.4706±0.04	*0.7245±0.02*
		Medium	**0.8570±0.03**	*0.7140±0.05*	0.6587±0.03	0.6095±0.12	0.4705±0.01	*0.7231±0.02*
		Complex	**0.8520±0.04**	*0.7110±0.03*	0.6571±0.04	0.6081±0.07	0.4703±0.02	*0.7230±0.01*

	Blended	**0.8510±0.05**	*0.7090±0.02*	0.6467±0.10	0.5980±0.03	0.4702±0.01	*0.7165±0.02*
UD	Simple	**0.8558±0.06**	*0.7132±0.07*	0.6560±0.06	0.6013±0.02	0.4704±0.03	*0.7163±0.01*
	Medium	**0.8546±0.07**	*0.7047±0.03*	0.6525±0.02	0.5965±0.10	0.4704±0.05	*0.7106±0.03*
	Complex	**0.8531±0.02**	*0.7045±0.02*	0.6487±0.04	0.5905±0.06	0.4700±0.02	*0.7100±0.01*
	Blended	**0.8526±0.03**	*0.7000±0.06*	0.6434±0.03	0.5901±0.06	0.4701±0.06	*0.7076±0.03*

4.6. Execution time comparison

In Table 14, we present the average computation time in seconds for 160 datasets (32 combinations × 5 datasets per combination) for each traffic accident dataset described in Table 7. The configuration of our machine is Intel® Core™ i5-3340M CPU @ 2.70GHz, with 8GB RAM. From this table, it can be seen that NoiseCleaner runs slightly faster than HARF-80, CPAD, and HARF-70. On the other hand, CAIRAD runs significantly faster compare to all the other approaches. As HARF-80 and HARF-70 are ensemble classifiers based on random forest and noisy instances are detected by using a cross validation strategy, they are computationally intensive. HCleaner and NoiseCleaner are both based on graph processing and are therefore computationally demanding as well. CPAD uses the GMM model to detect clusters, and optimizing the GMM model parameters is known to be very computationally intensive. On the other hand, CAIRAD is based on computing the co-appearance matrix and some simple statistics, and is therefore computationally fast. However, from our experimental results above, we can see that CAIRAD is also the poorest performing algorithm.

Table 14. Average execution time of different algorithms (in seconds)

Dataset name	NoiseCleaner	HARF-80	HARF-70	HCleaner	CAIRAD	CPAD
RCL	338.101	342.72	344.21	201.00	100.21	383.032
RC	165.21	184.01	183.88	178.10	50.34	231.645
MCI	95.34	99.92	96.68	64.55	35.65	287.412
MII	198.42	201.15	204.12	188.23	55.19	208.452

5. Conclusion

In many real world datasets, the attributes in a record are mostly categorical in nature. Detecting and correcting erroneous attribute values in categorical datasets is still a challenging problem since any attempt to numericalize a categorical value can introduce unwanted biases that negatively affect subsequent data analysis. In this paper, we have proposed a new noisy values identification and correction method, called NoiseCleaner, for traffic accidents data where the majority of attributes are categorical. In our approach, noisy attribute values are detected by using a novel *P-measure* which are probability values indicating the likeliness of replacing the noisy attribute value with some alternative attribute values, and the *S-measure* which measures the direct and transitive similarity between a noisy value and an alternative value to identifies the most similar alternative value to replace the noisy value with. Extensive experiments have shown that it outperforms several existing noisy values identification methods on four real world traffic accident datasets.

References:

[1] L.-Y. Chang and H.-W. Wang, "Analysis of traffic injury severity: An application of non-parametric classification tree techniques Accident analysis and prevention", Accident analysis and prevention, vol. 38, no. 5, pp.1019-1027, 2006.

[2] S.J. Delany, "The good, the bad and incorrectly classified: profiling cases for case-base editing", Proceeding of ICCBR conference, vol. 5650, pp. 135-149, 2009.

[3] R. Deb, A.W.C. Liew, "Missing value imputation for the analysis of incomplete traffic accident data", Information Sciences, vol. 339, pp. 274–289, 2016.

[4] R. Deb, A.W.C. Liew, E. Oh, "A correlation based imputation method for incomplete traffic accident data", Proceeding of PRICAI conference, vol. 8862, pp. 905–912, 2014.

[5] R. Deb, A.W.C. Liew, "Incorrect attribute value detection for traffic accident data", Proceeding of IJCNN conference, pp. 1-7, 2015.

[6] A. Farhangfar, L. Kurgan, J. Dy, "Impact of imputation of missing values on classification error for discrete data", Pattern Recognition, vol. 41, no.12, pp. 3692–3705, 2008.

[7] M. Fogue, P. Garrido, F.J. Martinez, J.-C. Cano, C.T. Calafte, "A novel approach for traffic accidents sanitary resource allocation based on multi-objective genetic algorithms", Expert Systems with Applications, vol. 40, no. 1, pp. 323-336, 2013.

[8] M.A. Hern´andez, S.J. Stolfo, "Real-world data is dirty: data cleansing and the merge/purge problem", Data Mining and Knowledge Discovery, pp. 9-37, 1998.

[9] R. Deb, A.W.C. Liew, "Missing Value Imputation for the Analysis of Incomplete Traffic Accident Data", Proceeding of ICMLC conference, pp. 275-286, 2014.

[10] A.W.C. Liew, N.F. Law, H. Yan, "Missing value imputation for gene expression data: computational techniques to recover missing data from available information", Briefings in bioinformatics, vol. 12, no. 5, pp. 498-513, 2011.

[11] X. Gan, A.W.C. Liew, H. Yan, "Microarray missing data imputation based on a set theoretic framework and biological knowledge", Nucleic Acids Research, vol. 34, no. 5, pp. 1608-1619, 2006.

[12] H. Junninen, H. Niska, k. Tuppurainen, J. Ruuskanen, M. Kolehmainen, "Methods for imputation of missing values in air quality data sets", Journal of Atmospheric Environment, vol. 38, no. 18, pp. 2895-2907, 2004.

[13] M.G. Rahman, M.Z. Islam, T. Bossomaier, J. Gao, "CAIRAD: A co-appearance based analysis for incorrect records and attribute-value detection", Proceeding of IJCNN conference, pp. 1-10, 2012.

[14] M.G. Rahman, M.Z. Islam, "kDMI: A Novel Method for Missing Values Imputation Using Two Levels of Horizontal Partitioning in a Data set", Proceeding of ADMA conference, vol. 8347, pp. 250-263, 2013.

[15] T. C. Redman, "The impact of poor data quality on the typical enterprise", Communications of the ACM, vol. 41, no. 2, pp. 79-82, 1998.

[16] B. Sluban, D. Gamberger, N. Lavra˘c, "Ensemble-based noise detection: noise ranking and visual performance evaluation", Data Mining and Knowledge Discovery, vol. 28, no. 2, pp. 265-303, 2014.

[17] C.M. Teng, "A comparison of noise handling techniques", Proceeding of AAAI conference, pp. 269-273, 2001.

[18] "Global status report on road safety", World Health Organization, Geneva, 2009.

[19] "Queensland Government data repository", https://data.qld.gov.au/dataset/crash-data-from-queensland-roads.

[20] "New York's open data portal", https://data.ny.gov/browse?tags=crash&utf8=%E2%9C%93.

[21] H. Xiong, G. Pandey, M. Steinbach, "Enhancing data analysis with noise removal", IEEE Transactions on Knowledge and Data Engineering, vol. 18, no. 3, pp. 304-319, 2006.

[22] X. Zhu, X. Wu, "Class noise vs. attribute noise: a quantitative study of their impacts", Artificial Intelligence Review, vol. 22, pp. 177-210, 2004.

[23] X. Zhu, X. Wu, Y. Yang, "Error detection and impact-sensitive instance ranking in noisy datasets", Proceeding of AAAI conference, pp. 378–383, 2004.

[24] E. Bigdeli, M. Mohammadi, B. Raahemi, S. Matwin, "A fast and noise resilient cluster-based anomaly detection", Pattern Analysis and Applications, vol. 20, no. 1, pp. 183-199, 2017.