Data Quality Enhancement for Traffic Accident Data

Rupam Deb

M.Sc. Engineering (Research based), B.Sc. Engineering (Honours)

School of Information and Communication Technology

Griffith Sciences

Griffith University

Submitted in fulfilment of the requirements of the degree of

Doctor of Philosophy

April, 2017

Abstract

Death, injury, and disability resulting from road traffic crashes continue to be a major global public health problem. Recent data suggest that the number of fatalities from traffic crashes is in excess of 1.25 million people each year with non-fatal injuries affecting a further 20-50 million people. It is predicted that by 2030, road traffic accidents will have progressed to be the 5th leading cause of death and that the number of people who will die annually from traffic accidents will have doubled from current levels. Both developed and developing countries suffer from the consequences of the increase in human population, and consequently, vehicle numbers. Therefore, methods to reduce accident severity are of great interest to traffic agencies and the public at large. To analyze traffic accident factors effectively, a complete traffic accident historical database is needed. Road accident fatality rates depend on many factors, so it is a very challenging task to investigate the dependencies between the attributes because of the many environmental and road accident factors. Missing data and noisy data in the database obscure the discovery of important factors and lead to invalid conclusions. In order to make the traffic accident datasets useful for analysis, they should be preprocessed efficiently. Data preprocessing is responsible for almost 80% of the total data mining effort. It is also known that good results can be achieved by using data mining algorithms only if there is a good quality dataset. This research is concerned with developing novel data preprocessing techniques for data quality enhancement, with application to traffic accident data. The research can be divided into two parts. The first part of this research concentrates on missing values imputation, and the second part concentrates on noisy values detection and correction in the traffic accident dataset. Missing values imputation and noisy values detection with correction are used to obtain a complete noise-free traffic accident dataset.

Statement of Originality

This work has not previously been submitted for a degree or diploma in any university. To the best of my knowledge and belief, the thesis contains no material previously published or written by another person except where due acknowledgement is made in the thesis itself.

_____

Rupam Deb

Acknowledgements

I would like to express my sincere gratitude and profound indebtedness to my supervisor Associate Professor Alan Wee-Chung Liew for constant guidance, insightful advice, helpful criticism, valuable suggestions, commendable support, and endless patience towards the completion of this research. I feel very proud to have worked with him. Without his inspiring enthusiasm and encouragement, this work could not have been completed. I appreciate all of his contributions which have made my PhD successful.

I also would like to extend my heartfelt gratitude to my associate supervisor Associate Professor Junhu Wang for his devoted guidance and sincere support during my PhD.

My special thanks are to my wife Chumky Das Gupta for her continuous encouragement and support in both the domestic and academic spheres of my life. I would also like to thank my little angels for their prayers and deep affection for me.

The support I have received from Dhaka University of Engineering and Technology in terms of approving study leave is gratefully acknowledged.

Last, but by no means least, I thank God for the talents and abilities I was given that made it possible to undertake this research.

Contents

List of Tables

List of Figures

Abbreviations

DMI [31]	Decision tree based Missing value Imputation
DSMI [17]	Decision tree and Sampling based Missing value Imputation
EMI [33]	Expectation Maximization Imputation
FIMUS [29]	Framework for Imputing Missing values Using co-appearance, correlation and Similarity analysis
kDMI [30]	k-Decision tree based Missing value Imputation
kNNI [20]	k-Nearest Neighbour based Imputation
LWLA [27]	Local Weighted Linear Approximation imputation
SCMI [2]	Sampling and Correlation based Missing value Imputation
SiMI [31]	Similarity based Missing value Imputation
FEMI [32]	Fuzzy Expectation Maximization and Fuzzy Clustering-based Missing Value Imputation Framework for Data Pre-processing
FCM [80]	Fuzzy C-Means
CPAANN [24]	CounterPropagation Auto-Associative Neural Network
GST [24]	Grey System Theory
GFCM [59]	General Fuzzy C-Means
FuzzyEM [33]	Fuzzy Expectation Maximization
RMSE [67]	Root-Mean-Square Error
EDIR [16]	Error Detection and Impact-sensitive instance Ranking
VIPER [72]	Visual Performance Evaluation of noise detection algorithms
HARF [72]	High Agreement Random Forest noise detection algorithm
RDCL [70]	Classify a record into Reachability, Dissimilarity, Coverage, Liability sets
CAIRAD [71]	Co-appearance based Analysis for Incorrect Records and Attribute-values Detection
NOISERANK [72]	Noise detection by ranking instances
HCleaner [74]	Hyperclique-based data Cleaner

List of Publications

01. Rupam Deb, Alan Wee-Chung Liew, “Missing value imputation for the analysis of incomplete traffic accident data”, Information Sciences, Vol. 339, pp. 274–289, 2016.
02. Rupam Deb, Alan Wee-Chung Liew, “Noisy values detection and correction of traffic accident data”, Information Sciences, January, 2017. (Under review)
03. Rupam Deb, Alan Wee-Chung Liew, “Missing value imputation for the analysis of incomplete traffic accident data”, Communications in Computer and Information Science, Springer, Vol. 481, pp. 275-286, 2014.
04. Rupam Deb, Alan Wee-Chung Liew, Erwin Oh, “A correlation based imputation method for incomplete traffic accident data”, Lecture Notes in Computer Science, Springer, Vol. 8862, pp. 905-912, 2014.
05. Rupam Deb, Alan Wee-Chung Liew, “Incorrect attribute value detection for traffic accident data”, In International Joint Conference on Neural Networks, pp. 1-7, Killarney, Ireland, July 2015.
06. Rupam Deb, Alan Wee-Chung Liew, “Resource allocation in accident prone areas by analysing the traffic accident data”, 13th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery, July 2017. (Accepted)
07. Rupam Deb, Alan Wee-Chung Liew, “Resource allocations by predicting the road crash severity instantly”, 13th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery, July 2017. (Accepted)

Chapter 1: Introduction

At the current time, modern organizations collect a large amount of data due to the advances in information processing technology and storage capacity. To extract knowledge from the large volume of a dataset using data mining techniques, there is a strong need for data preprocessing algorithms to ensure the data is of good quality. It is well known that good results can only be achieved in data mining if the dataset is of good quality [1]. Real world data is highly susceptible to missing values, noisy values, and inconsistencies. Data preprocessing is a key step that processes the raw data into a form that facilitates subsequent analysis. This research presents two data preprocessing techniques to enhance the quality of data: (i) DSMI algorithm to impute the missing values and (ii) NoiseCleaner to remove the noisy values from the traffic accident datasets.

This chapter is organized into several sections as follows. Section 1.1 presents an overview of the research. Section 1.2 describes the data preprocessing concept. Then, the research objectives and research questions are described in Section 1.3. Section 1.4 presents the research contributions made in this thesis. Finally, the structure of this thesis is presented in Section 1.5.

1.1 Overview

An accident is an unplanned and unwanted event which disrupts the work process and causes injury to people. A traffic accident occurs when a vehicle collides with another vehicle, pedestrian, animal, road debris, or other stationary obstruction, such as a tree or utility pole. Death, major or minor types of injuries, vehicle, and property damage are the result of traffic collisions.

Accident investigations and analyses are performed to determine the causes of an accident. Accidents of particularly common types, such as automobile crashes are often investigated to identify how to avoid them in the future. The accident investigation is often performed by a range of experts, including forensic scientists, forensic engineers or health and safety advisors.

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 of traffic accidents on the road, which has become a serious problem in many countries. For example, 2478 people died on Spanish roads in 2010, which means one death for every 18,551 inhabitants [2-4]. In the United States (according to the Department of Transportation, United Sates) in 2012, 33,561 people died in motor vehicle traffic crashes [5]. According to the Australian Bureau of Statistics, the majority of transport related deaths (72% in 2009) in Australia is associated with motor vehicles driven on public roads [6-8]. The global economic cost of road traffic accidents has been estimated at US$518b and has been calculated to account for 0.3% to 4% of the gross national product of many countries [9].

The issue of traffic safety has become one of the key challenges in the sustainable development of modern traffic and transportation. The burden of road accident causalities and damage is a major headache for both developed and developing countries. Motor vehicle collisions lead to loss of lives and permanent disabilities, and incur large financial costs to both the community and the individuals involved. There are many factors that contribute to the risk of collisions such as vehicle design, vehicle speed, road design, road environment, driver skill, and behaviour. Therefore, it is essential for traffic engineers to be able to extract useful knowledge from existing data to analyse the causes of traffic accidents and to determine the factors which affect the severity of injuries in road crashes. Such information enables traffic administrators to be more accurately informed such that better policies can be introduced to reduce the number of road traffic accidents.

A large amount of traffic accident data is stored in various types of databases because of the advancement of data acquisition methods and storage technology. The

improvement in sensor technologies has also resulted in the growth of large amounts of traffic accident data [10]. We can extract traffic activities and factors which lead to traffic accidents from the traffic accident databases by using advanced data mining technology. Data mining is typically conceptualized as a three part process: preprocessing, learning and post-processing. This thesis focuses on the preprocessing stage, as this can take up to 80% of the total data mining effort [11].

1.2 Data preprocessing

Data preprocessing is a technique which involves transforming raw data into a format that is suitable for subsequent analysis. It is well known that good results can only be obtained from data mining algorithms if there is a good quality dataset [1]. Real world datasets often have missing values and noisy values due to various reasons, including equipment malfunction, human error, and faulty data transmission. If an organization does not take extreme care during data collection, then large amounts of missing and noisy data could be introduced into the datasets [12-16]. Data preprocessing tasks can include the imputation of missing values, smoothing of noisy data, identification of erroneous data, and correction of erroneous data.

Missing value means the data value is missing for the variable in the dataset. Many applications in the real world suffer from a common problem that some values of the attributes are unobserved. Table 1.1 shows a toy dataset to explain missing values. In this dataset, '?' stands for missing value. Missing values can be numerical or categorical 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 connecting scale (e.g., age set 13-19).

Table 1.1. Toy dataset to explain missing values

Record	Driver status	Weather condition	Passenger number	Accident address	Injury severity
R_1	Drunk	Good	3	Sanders	Kill
R_2	Drunk	Good	4	?	Kill
R_3	Drunk	Good	2	Glendale	No injury
R_4	Normal	Fair	3	Glendale	No injury
R_5	Normal	Fair	?	Glendale	No injury
R_6	?	Good	?	Glendale	Kill

Noisy data means corrupted or meaningless data. Noisy data badly affect the results of any type of data analysis. The presence of noisy values reduces the quality of the analysis models learned from the data and impairs their predictive or descriptive performance. Moreover, these models would become overly complex if the noise in the data were to be accommodated.

Many approaches have been proposed to deal with missing values in a dataset. Some of the popular approaches are:

- Discarding the record: this is generally used when the class label is lost. This technique performs well when the dataset has few records with missing attributes. However, discarding records throws valuable data away and can hamper efforts to extract knowledge from the data, especially when the dataset is small. It also introduces bias into the dataset when the proportion of missing values per attribute varies significantly in the dataset.
- Replacing the missing value by a universal constant: this method replaces all missing attribute values by a constant, for example: “missed” or “?” or zero. The main problem is that when all missing values are replaced for example by “missed”, then the data mining algorithm mistakenly concludes that they form an interesting concept, since they all have a value in common, that of

"missed". Therefore, although this method is simple, it is not recommended for data cleaning.

- Impute the missing value by the mean attribute value: this introduces bias into the dataset. Moreover, it is infeasible when imputing categorical missing values.

1.3 Research objectives and research questions

Two important data preprocessing tasks are missing value imputation and data cleansing. This research proposes effective algorithms to perform these tasks for traffic accident data. The objectives of this research are:

I. To impute the missing values for the traffic accident datasets, where a large number of attributes values are categorical in nature.

II. To clean noisy values in traffic accident datasets.

Although many algorithms have been proposed for missing value imputation, most of these algorithms are developed for missing numerical data. However, in many real world datasets, many attributes are categorical. In fact, most attributes in the traffic accident datasets are categorical in nature. Very little research has been undertaken to handle missing categorical attributes. The same is true for data cleansing, where most existing algorithms are for numerical datasets. The above objectives have therefore led to the following research questions:

I. How can missing value imputation algorithms handle categorical attributes?

II. How can the correlation between categorical attributes be measured and quantified?

III. How can we measure correlation between two records with categorical values?

IV. How can we take into account the uncertainty in attribute values seen in real data?

V. What criteria can be used to detect noisy categorical data?

1.4 Contributions

The main research contributions made in this thesis are presented below:

I. We proposed an effective missing values imputation algorithm called DSMI for traffic accident data, where a large number of attributes values are categorical.

II. DSMI algorithm is able to consider the inherent uncertainty seen in real data.

III. We proposed a novel noisy values identification and correction method, called NoiseCleaner, which can identify noisy categorical and numerical attributes values in the traffic accident datasets.

IV. We performed extensive experiments to evaluate the performance of the proposed algorithms and compare them with current state-of-the-art algorithms.

1.5 Thesis structure

This thesis is divided into four chapters. Chapter 1 presents the introduction for this research, data preprocessing concepts, motivation, research objectives, research questions, research contributions, and the thesis structure. Chapter 2 focuses on the first part of the research, i.e. missing value imputation. It reviews the literature relevant to

missing value imputation and then presents the proposed missing value imputation algorithm called DSMI. Experimental results and comparative studies with existing missing value imputation algorithms are also given in this chapter. Chapter 3 presents the second part of the research, i.e. data cleansing. It reviews the literature in this area and presents the proposed noisy values detection and correction algorithm called NoiseCleaner. Extensive experimental results and comparative studies on data cleansing are also reported. Finally, Chapter 4 provides the conclusions and recommendations for further study. The structure of the thesis is shown in Figure 1.1.

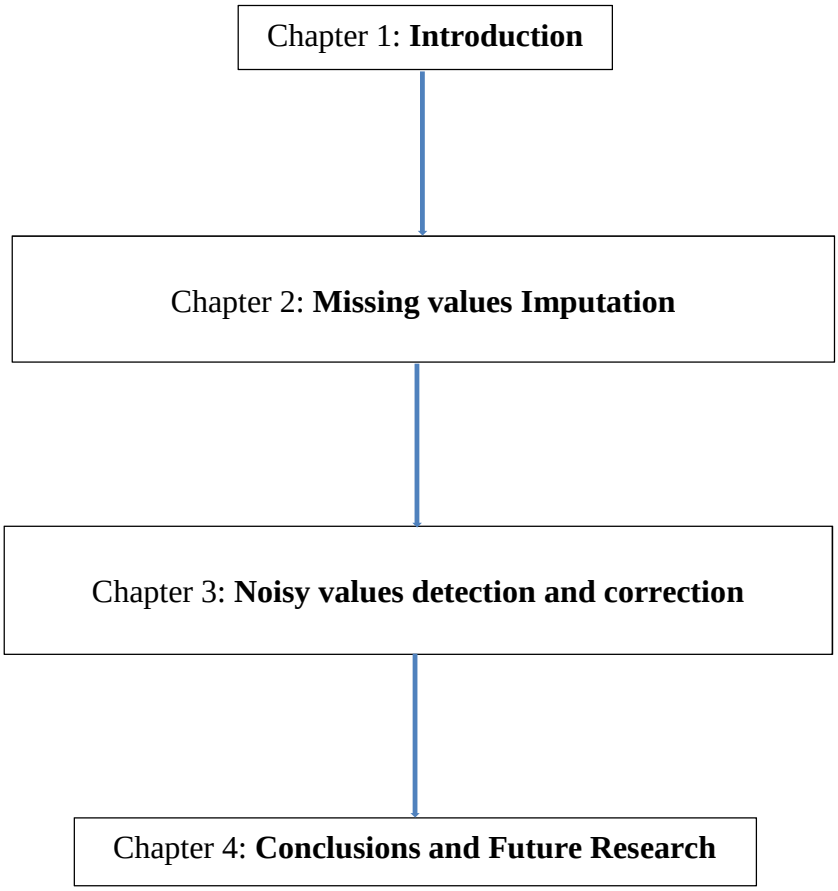


Figure 1.1: Thesis structure

Chapter 2: Missing values Imputation

This chapter describes the first part of the research, which is on missing values imputation. An introduction about missing value imputation is given in Section 2.1. In Section 2.2, some existing missing values imputation approaches are reviewed and analysed. Section 2.3 presents the proposed missing value imputation algorithm called Decision tree and Sampling based Missing values Imputation (DSMI) in detail. Section 2.4 addresses the performance of the DSMI algorithm. A summary is given at the end of this chapter. The content of this chapter has largely been published in [17].

2.1 Introduction

Huge quantities of data are collected every day from sources such as surveys, interviews, Facebook, Twitter, LinkedIn, and sensors [18]. For example, the habits and profiles of people are collected from Facebook and Twitter, professional data are collected from LinkedIn, and weather data in an environment monitoring system are often acquired through different sensors. Data can be missing or noisy due to numerous causes such as equipment malfunctioning and errors incurred during data alteration and transmission. In an environment monitoring system, data can be missing due to limited bandwidth in a wireless network, inadequate battery power of the sensing devices, other hardware, and software problems in the sensors.

To extract useful information from traffic accident datasets, we need a complete dataset. Missing data are a common occurrence in real world data collection and have a significant impact on the conclusions that can be drawn from the data. The main problem of missing values is that the analysis is impossible or distorted because of the missing values. To overcome this problem, researchers design appropriate protocols to minimize the occurrence of missing values and develop effective imputation algorithms

to impute the missing values. The imputation of missing values as accurately as possible is an important data preprocessing task.

In this chapter, a novel method called Decision tree and Sampling based Missing values Imputation algorithm DSMI [17] is presented and compared with other imputation algorithms.

2.2 Literature review

Imputation of missing values is an important data preprocessing task for improving the quality of the data. Many missing value imputation algorithms have been proposed for various applications [19-54]. Some of these methods are: Expectation Maximization Imputation (EMI) [33], Decision tree based Missing value Imputation (DMI) [31], combined instance selection with K-Nearest Neighbour Imputation [34], Similarity based Missing value Imputation (SiMI) [31], k-Decision tree based Missing value Imputation (kDMI) [30], k-Nearest Neighbour based Imputation (kNNI) [20], Local Weighted Linear Approximation imputation (LWLA) [27], Framework for Imputing Missing values Using co-appearance, correlation and Similarity analysis (FIMUS) [29], and Fuzzy Expectation Maximization and Fuzzy Clustering-based Missing value Imputation (FEMI) [32].

To impute numerical missing values, the EMI algorithm [33] relies on estimating the mean and covariance matrix of the dataset. The EMI algorithm begins with an initial estimate of the mean and the covariance matrix, and iterates until the imputed values and the estimates of the mean and covariance matrix stop changing appreciably from the current to the next iteration [33, 55]. The EMI algorithm is only applicable to datasets in which the missing values are missing at random. The main drawback of this method is that for imputing the missing value, the EMI algorithm uses information from the whole dataset and therefore is suitable only for datasets that exhibit strong correlations between attributes. A Fuzzy c-Means (FCM) algorithm is proposed to impute the numerical missing data in [90].

Instead of using information from the whole dataset, the kNNI method [20] imputes missing values using k number of similar records. This method first finds user-defined k number of records from the total dataset by using the Euclidean distance measure. To impute a numerical missing value, the method utilizes the mean value of the specific attribute within the k most similar records of the entire dataset. If the missing attribute is categorical, then the method utilizes the most frequent value of the attribute within the k most similar records. The simple kNNI method performs well on a dataset that has strong local correlation structure. However, the method can be expensive for large dataset since for each record with missing value(s), it finds k number of similar records by searching the whole dataset. This is the main drawback of this method, especially for large datasets.

Rahman et al. proposed the DMI [31] technique which uses the decision tree and the EMI algorithm for missing value imputation. They argued that the correlations among attributes within a horizontal partition of a dataset can be higher than the correlations over the whole dataset. This technique works as follows: it first divides the full dataset (D_{Full}) into two sub-datasets, one having records with missing values (D_{Miss}), and the other having records without missing values ($D_{Complete}$). Then, it builds decision trees on $D_{Complete}$ considering the attributes having missing values in D_{Miss} as class attributes. After that, it assigns each record with missing value(s) in D_{Miss} to the leaf where it falls in for the tree, which considers the attribute that has a missing value for the record as the class attribute. Finally, it imputes numerical missing values using the EMI algorithm and categorical missing values using majority class values within the leaves. The authors showed that DMI performed well compared with other existing imputation methods. However, for imputing categorical values, simple voting is used. Another more serious problem is that the authors do not define how the imputation is done if the missing values record falls in more than one leaf; a situation that could occur if there is more than one missing value in a record.

SiMI [31] is an extension of DMI. It uses the decision forest algorithm to identify horizontal segments of a dataset where the records belonging to a segment have higher similarity and attribute correlations. It also divides the whole dataset (D_{Full}) into

$D_{Complete}$ and D_{Miss}. Then, it builds a decision forest using $D_{Complete}$. The decision forest builds a number of trees with leaves and assigns each record of D_{Miss} to the most suitable leaf. After that, SiMI finds the intersections of the records belonging to the leaves of the forest. Then, it imputes numerical missing values using the EMI algorithm and categorical missing values using the most frequent values.

The kDMI [30] algorithm imputes missing values using two levels of partitioning. Like DMI, the kDMI algorithm also employs horizontal partitioning based on a decision tree in the first level partitioning. For the second level partitioning, the authors use a BestKNN approach to first find the best value of k by searching all the records of a leaf and calculating the root mean square error (RMSE) of the non-missing attribute values. Then, the EMI algorithm is used for imputing numerical data, and the frequent value of BestKNN is used for imputing categorical data. However, what is not clear is that if all the attributes of a record are categorical, then there is difficulty in knowing how the RMSE can be calculated using BestKNN. It is also not clear how the imputation would be done if the missing values record falls in more than one leaf.

The FIMUS technique [29] takes the imputation decision based on the co-appearances of the values, the correlations between attributes, and similarity of values belonging to an attribute. In this method, the authors show that it is possible to achieve a better imputation result by considering available attributes values and their similar values. Here, similarity (1st level and 2nd level similarities) is calculated using the co-occurring of attribute values between the records of a dataset [56]. The 1st level similarity is calculated using the co-occurrence of attribute values of the records, and the 2nd level uses the “neighbours of neighbours are the direct neighbours” method. This algorithm uses similarity and co-appearance at the same time. The main problem of this algorithm is its computational complexity. If the number of records in the dataset is increased, then it needs massive computation for similarity graphs. Another problem is that to impute values, similarity values depend on the co-appearance values (in FIMUS, similarity value is multiplied by the co-appearance value). If there is no co-appearance value, then the associated similarity value has no impact to impute missing values. They validate the imputation accuracy using the RMSE and Index of agreement. However, it

is not known how validation can be done for categorical attributes values using these evaluation criteria.

The Genetic Algorithm (GA) is also used to impute missing values. The Fuzzy C-means (FCM) algorithm exploits GA to estimate the set of parameters [19]. This technique imputes the missing values two times. During the first time, the missing value is imputed using support vector regression and during the second time using FCM with a set of user-defined parameters. Then, these two sets of imputed values are compared to test their mutual agreement. If they are similar, then the missing value imputation is stopped otherwise the GA algorithm is applied to re-estimate the parameters, and this process is continued until these two parameters are not similar.

Gautam et al. proposed two imputation methods: the CounterPropagation Auto-Associative Neural Network (CPAANN) based imputation method and a hybrid of CPAANN along with Grey System Theory (GST) i.e. Grey + CPAANN imputation method [24]. There are three layers in the CPAANN: the input layers, the hidden layers, and the output layers. The input nodes are fully connected to the hidden nodes and the hidden nodes, in turn, are fully connected to the output nodes. Additionally, all the hidden nodes are connected to one another. In the CPAANN model, the given dataset is divided into two parts: the complete dataset and the incomplete dataset. This model is trained using the complete dataset. After that, it makes the mean imputation on the incomplete dataset and sends it to the trained CPAANN for performing the test using these mean imputed incomplete records. The Grey + CPAANN model is almost the same as the CPAANN model. The given dataset is divided into complete and incomplete datasets. After that, it trains the CPAANN model using the complete dataset. The difference between Grey + CPAANN and CPAANN is that instead of mean imputation, the Grey + CPAANN method makes the distance based nearest neighbour imputation [57]. Then, testing is performed as per the CPAANN model.

In many real world datasets, all data are not equally informative and some data points can be considered as noisy points. Instance selection means some data points are removed, but the integrity of the original dataset can still be maintained. Tsai et al. [34] proposed a missing value imputation algorithm combined instance selection with k-

Nearest Neighbour imputation (kNNI). The authors present four approaches by combining instance selection and imputation. In the first combination process, imputation is performed first and then instance selection is used to reduce the dataset. In this approach, the training dataset (D) is divided into two parts: the complete dataset (D_complete) and the incomplete dataset (D_incomplete). Then, using the kNNI, the imputation is done into D_incomplete and produces a new complete training dataset D'. Next, the instance selection algorithm is executed to remove the unusual instances from D' and makes a new reduced dataset called D1. Finally, a classifier is trained by D1 and tested by a given testing dataset to examine the classifier performance. In the second combination, instance selection is performed first and missing value imputation is done next. Instance selection is performed over D_complete to make a new complete reduced dataset D_complete'. After that, the imputation is performed over D_incomplete and subsequently, a new complete dataset D2 is produced. Finally, a classifier is trained by D2 and tested by a given testing dataset to examine its classifier performance. The third and fourth combination processes are based on a two-stage instance selection process. In the third combination, instance selection is performed again over D1 and a new reduced training dataset D3 is obtained. In the fourth combination, instance selection is performed again over D2 and a new reduced training dataset D4 is obtained. The authors determine the missing value imputation accuracy by the classification accuracy of the K-NN and the linear SVM classifiers. The Drop3 algorithm [58] is used to perform the instance selection.

A Fuzzy Expectation Maximization and Fuzzy Clustering-based Missing value Imputation (FEMI) algorithm is proposed by Rahman et al. [32] to impute the numerical and categorical attributes values. This algorithm works as follows: at first, all numerical attributes of the dataset are normalized to be within a range between 0 and 1. Then, the dataset is divided into two parts: D_C (having no missing values) and D_I (having missing values). In the next step, it finds the membership degrees of all records of D_C and D_I with all clusters using the General Fuzzy C-Means (GFCM) algorithm [29]. To impute the numerical missing values the authors proposed a Fuzzy Expectation Maximization (FuzzyEM) algorithm. This FuzzyEM algorithm is a modification of an existing EM

algorithm [33]. The idea of this algorithm is that the missing values are imputed using the mean of the membership degrees of all clusters. To impute the categorical missing values of the cluster, a vote is taken for all domain values of this cluster and the value having the maximum vote is considered to the imputed value. The vote is calculated for a value by multiplying its confidence degree in terms of its cluster and the membership degree. The confidence degree of an attribute value in a cluster is the sum of the membership degrees for the records, having this attribute value. The authors determine the missing value imputation accuracy by using two evaluation criteria: RMSE and Mean Absolute Error (MAE). However, it is not known how validation can be done for categorical attribute values using these two evaluation criteria.

An automated data imputation model based on a three layered artificial neural network is used for missing value imputation [58]. Here, the numbers of neurons in both the input and output layers are equal to the number of attributes of the full dataset. By taking some available values as missing values, the neural network is trained. This automated data imputation model finally imputes the missing values using the trained network.

2.3 Proposed DSMI algorithm

The proposed DSMI algorithm [17] is based on decision trees. In DSMI algorithm, similar to other decision tree-based imputation algorithms, two datasets are created from the original dataset. The first dataset, denoted as the complete dataset, contains records with no missing values. The second dataset, denoted as incomplete dataset, contains records with one or more attributes values missing, i.e. we called them missing records. Then, for each missing attribute, a decision tree that uses the missing attribute as class attribute is constructed from the complete dataset. Each missing record is then assigned to the corresponding tree's leaf. Once a missing record is assigned to a leaf node, the missing values in the missing record are imputed using records that are found in the leaf node.

The algorithm is illustrated here with a toy example (? stands for missing value). Table 2.1 shows a full dataset D_{Full}. The full dataset (D_{Full}) is first divided into two sub-datasets. One subset contains records with missing values (D_{Miss}) and the other without missing values ($D_{Complete}$). If the value of an attribute is numerical, we quantize it by the square root of its domain size. Tables 2.2-2.3 show the resulting $D_{Complete}$ and D_{Miss} respectively. Next, a set of decision trees with class attributes given by the attributes having missing values in D_{Miss} are built using C4.5 [60, 61] algorithm using records from $D_{Complete}$. For example, three attributes in D_{Miss}, i.e. Driver status, Passenger number, and Accident address, have missing values, and three decision trees are created based on these class attributes, as shown in Figures 2.1-2.3.

Table 2.1. Full dataset D_{Full}

Record	Driver status	Weather condition	Passenger number	Accident address	Injury severity
R_1	Drunk	Good	3	Sanders	Kill
R_2	Drunk	Good	4	?	Kill
R_3	Drunk	Good	2	Glendale	No injury
R_4	Normal	Fair	3	Glendale	No injury
R_5	Normal	Fair	?	Glendale	No injury
R_6	?	Good	?	Glendale	Kill

Table 2.2. Complete dataset $D_{Complete}$ with quantized passenger number

Record	Driver status	Weather condition	Passenger number	Accident address	Injury severity
R_1	Drunk	Good	3-4	Sanders	Kill
R_3	Drunk	Good	1-2	Glendale	No injury
R_4	Normal	Fair	3-4	Glendale	No injury

Table 2.3. Missing value Dataset D_{Miss} with quantized passenger number

Record	Driver status	Weather condition	Passenger number	Accident address	Injury severity
R_2	Drunk	Good	3-4	?	Kill
R_5	Normal	Fair	?	Glendale	No injury
R_6	?	Good	?	Glendale	Kill

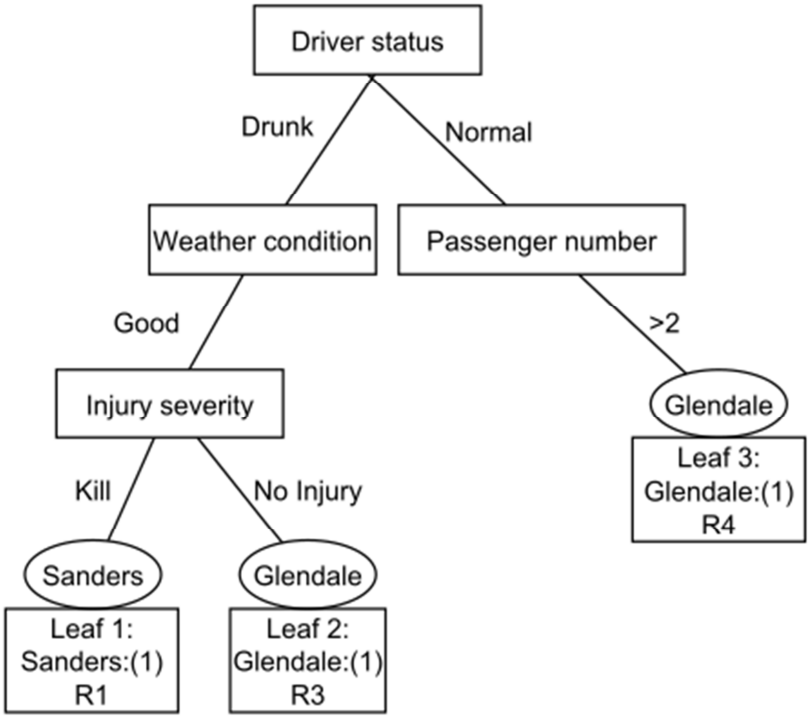


Figure 2.1. Tree for ‘Accident address’ class

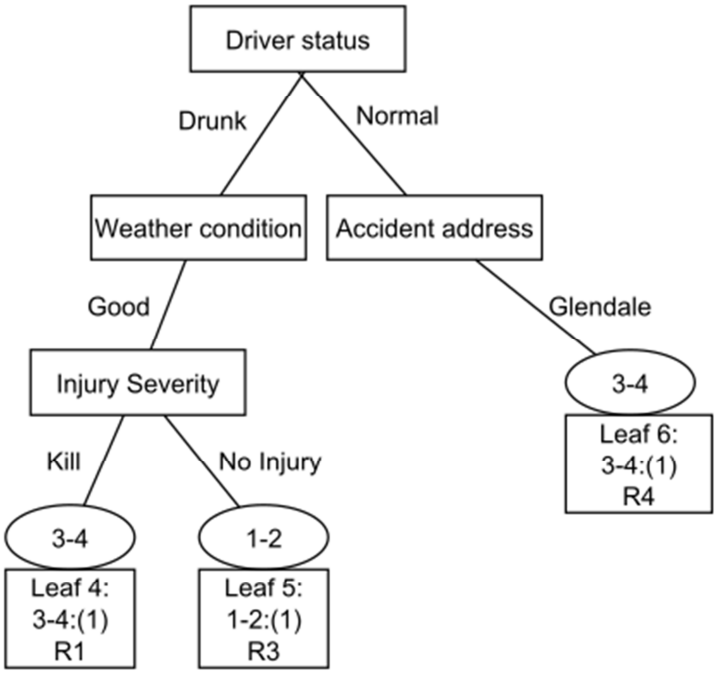


Figure 2.2. Tree for ‘Passenger number’ class

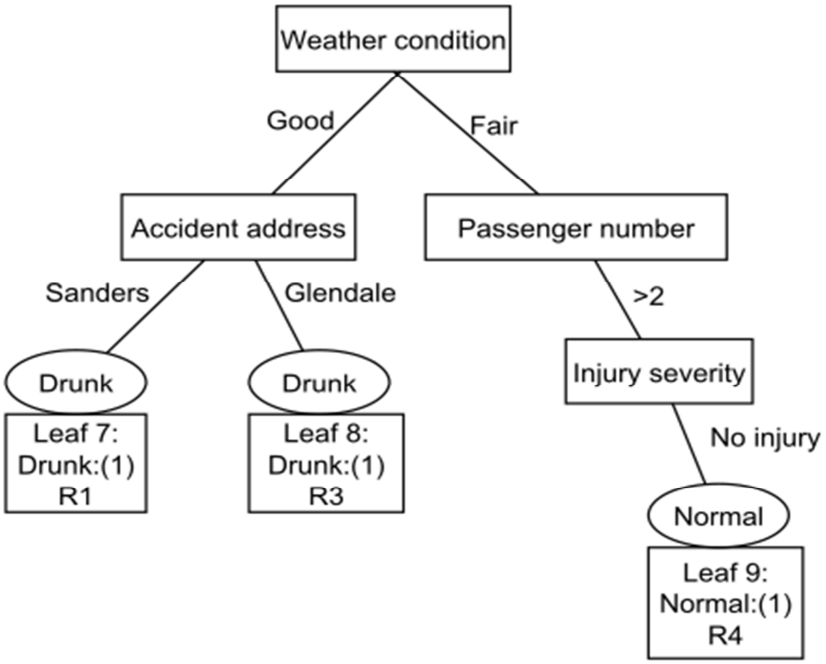


Figure 2.3. Tree for ‘Driver status’ class

In Figure 2.1, 'Leaf 1 Sanders: (1) R_1' denotes one record R_1 falls in Leaf 1 which is associated with the attribute value 'Sanders'. After the tree construction step, we assign each record in D_{Miss} to the leaf of the tree with the same class attribute as the missing attribute. For example, R_2 record has 'Accident address' attribute value missing so we assign it to the 'Accident address' tree. Therefore, R_2 and R_5 records are assigned to Leaf 1 and Leaf 6, respectively. Records with more than one missing values would fall into multiple leaves. As R_6 record has two missing values, it is assigned to three leaves 4, 6, and 8. Once all records in D_{Miss} are assigned to the appropriate leaves, each leaf will consists of records from $D_{Complete}$ and D_{Miss} that are correlated. Tables 2.4-2.7 show the sets of records in leaves with missing records. If a record falls into multiple leaves, the records from all these leaves are grouped into one collection. Therefore, the set of records associated with R_6 are given by Table 2.8.

Table 2.4. Records in leaf 1

Record	Driver status	Weather condition	Passenger number	Accident address	Injury severity
R_1	Drunk	Good	3	Sanders	Kill
R_2	Drunk	Good	4	?	Kill

Table 2.5. Records in leaf 4

Record	Driver status	Weather condition	Passenger number	Accident address	Injury severity
R_1	Drunk	Good	3	Sanders	Kill
R_6	?	Good	?	Glendale	Kill

Table 2.6. Records in leaf 6

Record	Driver status	Weather condition	Passenger number	Accident address	Injury severity
R_4	Normal	Fair	3	Glendale	No injury
R_5	Normal	Fair	?	Glendale	No injury
R_6	?	Good	?	Glendale	Kill

Table 2.7. Records in leaf 8

Record	Driver status	Weather condition	Passenger number	Accident address	Injury severity
R_3	Drunk	Good	2	Glendale	No injury
R_6	?	Good	?	Glendale	Kill

Table 2.8. Aggregated table for record 6 with multiple missing values

Record	Driver status	Weather condition	Passenger number	Accident address	Injury severity
R_1	Drunk	Good	3	Sanders	Kill
R_4	Normal	Fair	3	Glendale	No injury
R_3	Drunk	Good	2	Glendale	No injury
R_5	Normal	Fair	?	Glendale	No injury
R_6	?	Good	?	Glendale	Kill

To impute the missing values in the missing record, we search for records in the table which have the maximum number of non-missing attributes in common to the missing record. Then, the attribute values in these records corresponding to the missing

attributes in the missing record are taken to be the possible imputed values. For example, in Table 2.8, R_6 record has three non-missing values but we do not get any record matching with these three non-missing values. So, we search instead for two matching non-missing values and get two records with two matching attributes values: R_1(Good, Kill) and R_3(Good, Glendale). For the two missing attributes (Driver status, Passenger number), the possible imputed values from R_1 and R_3 records are (Drunk, 3), and (Drunk, 2), respectively. To decide which possible imputed values are more likely for the missing record, we need a measure of the affinity of possible imputed values to the missing record. Let ϑ denote the affinity degree of the possible imputed values for the missing record. The next step is to evaluate the affinity degree ϑ for each of the two possible imputed values (Drunk, 3), and (Drunk, 2).

In our approach, we use two measures to exploit the correlations between records and between attributes values within a record to impute the missing values. The first measure, called IS measure, computes the correlations between attribute values of different attributes in a record. The second measure, called weighted similarity measure (S_{ij}), computes the similarity between two values of an attribute in two different records. These measures are computed as follows.

2.3.1 IS measure

We measure the correlation between set of attributes with non-missing values, C, and set of attributes with missing values, M, within a record using *IS* measure. The *IS* measure measures the degree of associations between two sets of attributes values [62, 63]. $Let, C = [A_1, A_2, \dots, A_n]$, and $M = [B_1, B_2, \dots B_m]$, where $A_1, A_2, \dots A_n$ are the attributes with non-missing values and $B_1, B_2, \dots B_m$ are the attributes with missing values, then

$$IS(C, M) = \frac{Support(C, M)}{\sqrt{Support(C) \times Support(M)}} \quad (1)$$

where $support(C, M) = |C, M|/Q$, and$|C, M|$ denotes the number of records that contain both C and M, and Q refers to the size of the dataset.

IS measure has several desirable properties [62, 64]. It is the product of two important quantities: interest factor and support count. Interest factor *I* is a popular measure for association like patterns [45, 46] and is defined to be the ratio between the joint probability of two variables x and y with respect to their expected probabilities under the independence assumption. Let x and y denote a pair of binary variables, then *I* is defined by

$$I(x, y) = \frac{P(x,\ y)}{P(x)\ P(y)} \qquad (2)$$

The dataset that contains these variables can be summarized into a 2×2 contingency table as shown in Table 2.9. Each cell represents the four possible combinations of x and y values. f_{ij} corresponds to the support count for each cell; while f_{i+} and f_{+j} are the marginal sums of row i and column j (for example, $f_{i+} = f_{i1} + f_{i0}$). Then, it can be shown that

$$I\ (x, y) = \frac{f_{11} Q}{f_{1+} f_{+1}} \qquad (3)$$

Table 2.9. 2×2 contingency table for binary variables

	y	$\bar{y}$	
x	f_{11}	f_{10}	f_{1+}
$\bar{x}$	f_{01}	f_{00}	f_{0+}
	f_{+1}	f_{+0}	Q

However, *I* in itself could lead to association rule that is counter-intuitive due to not considering the support of the association, as pointed out in [11, 52]. Instead, the *IS* measure *IS*(*x*, *y*) takes into account *Support* (*x*, *y*) = f_{11}/Q:

$$IS\ (x,y) = \sqrt{I(x,y) \times \frac{f_{11}}{Q}} \qquad (4)$$

In addition, *IS* measure is equivalent to the geometric mean of confidence of rules that can be generated from the item-pair i.e. $IS\ (x,y) = \sqrt{confidence(x\ ,\ y) \times confidence(y\ ,\ x)}$ and this measure does not depend on data size. Here,$confidence\ (x,y) = P\ (x\ ,y)/P\ (x)$, with $P\ (x) = |x|/n$ and $|x|$ is the number of transactions that contain x and n is the total number of transactions.

2.3.2 Weighted similarity measure

To evaluate how similar two records in the dataset are, we use the weighted similarity measure, S_{ij}. This weighted similarity measure considers both the direct relationship (called 1st level similarity) and transitive relationship (called 2nd level similarity) between two attribute values of an attribute. The first step in calculating S_{ij} between two attribute values of an attribute is to create a graph. Let *G* = (*V*, *E*) be a graph created from the dataset of related records found in the leaf of a tree, i.e. the set of records in a table, where the set of vertices *V* are given by the set of attribute values in the dataset and an edge $e \in E$ *is* drawn between two vertices when both attribute values appear in a record. Then, S_{ij} is the weighted sum of the S'_{ij} (1st level similarity) and S''_{ij} (2nd level similarity) between two attribute values of an attribute

$$S_{ij} = C_1 \times S'_{ij} + C_2 \times S''_{ij} \qquad (5)$$

where C_1 and C_2 are weights with $C_1 + C_2 = 1$.

The 1st level similarity, S'_{ij} is given by

$$S'_{ij} = \frac{\sum_{k=1}^{n} \sqrt{a_{ik} \times a_{kj}}}{\sqrt{d(i) \times d(j)}} \quad (6)$$

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 and $d(i)$ is the degree of vertex i. Note that S'_{ij} effectively measures the proportion of common neighbours of vertices v_i and v_j against all direct neighbours of the two vertices. If the two vertices have all their neighbours in common, then S'_{ij} has a maximum value of 1. If the two vertices do not share any common neighbours, then S'_{ij} is zero.

On the other hand, the 2nd level similarity, S''_{ij} measures the transitive relationship for two attribute values of an attribute that are not directly connected to common neighbours but are connected to pair of attribute values v_p and v_q who have 1st level similarity S'_{pq} greater than a user defined threshold T. The 2nd level similarity S''_{ij} is equal to the 1st level similarity between v_i and v_j computed from the merged graph where v_p and v_q are merged into a common vertex.

Finally, the weighted similarity measure $S(R_k, R_l)$ between two records R_k and R_l is given by averaging the weighted similarity measure S_{ij} of each attribute of the two records [17, 36].

To illustrate how *IS* and S_{ij} are computed, we will use the records in Table 2.8 (presented in Table 2.10 with vertex nodes labelled from 1 to 10) as an example. Recall that for the two missing attributes (Driver status, Passenger number) in R_6, the possible imputed values from R_1 and R_3 records are (Drunk, 3), and (Drunk, 2), respectively. The *IS* measure for the two records are

$$IS_{R_1}[(Good, kill), (Drunk, 3)] = \frac{support(Good, kill, Drunk, 3)}{\sqrt{support(Good, kill) \times support(Drunk, 3)}} = 1$$

and $IS_{R_3}[(Good, Glendale), (Drunk, 2)] = 1$.

Table 2.10. Aggregated table for record 6 with vertex node number assigned

Record	Driver status	Weather condition	Passenger number	Accident address	Injury severity
R_1	Drunk (1)	Good (3)	3 (5)	Sanders (7)	Kill (9)
R_4	Normal (2)	Fair (4)	3 (5)	Glendale (8)	No injury (10)
R_3	Drunk (1)	Good (3)	2 (6)	Glendale (8)	No injury (10)
R_5	Normal (2)	Fair (4)	?	Glendale (8)	No injury (10)
R_6	?	Good (3)	?	Glendale (8)	Kill (9)

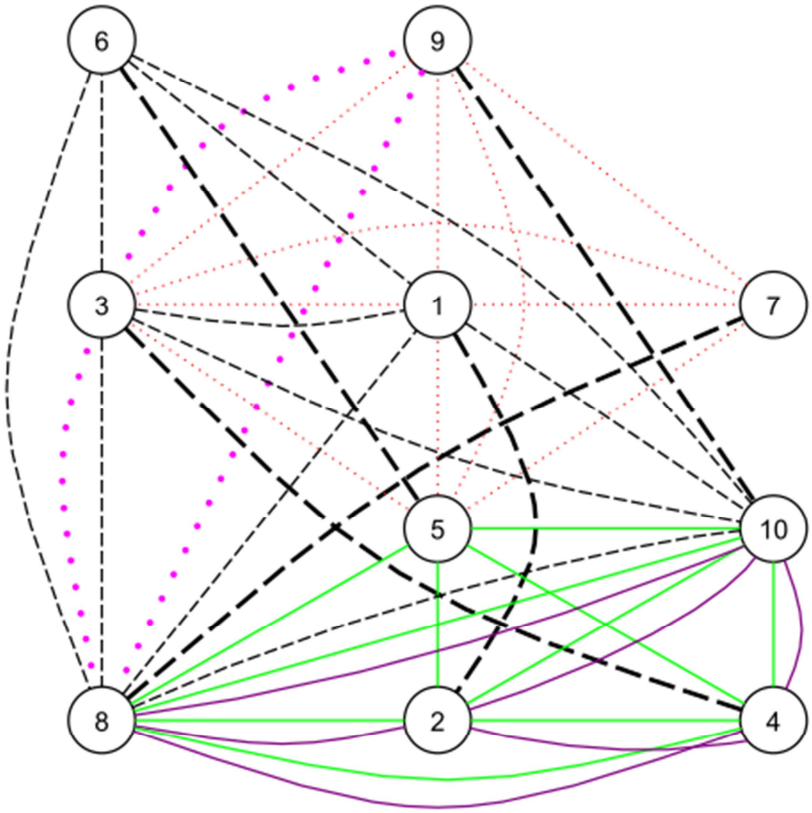


Figure 2.4: Graph constructed from Table 2.10

Next, to compute the weighted similarity measure between two records, we need to first calculate S_{ij} for R_6 from Table 2.10 and the corresponding graph in Figure 2.4. From the graph in Figure 2.4, we can see that nodes 7 and 8 have nodes 1, 5, 9, 3 as common neighbours, hence the 1st level similarity between nodes 7 and 8 is $S'_{7,8} = \frac{\sqrt{(1\times1)}+\sqrt{(1\times1)}+\sqrt{(1\times1)}+\sqrt{(1\times2)}}{\sqrt{4\times13}} = 0.613$. Similarly, we can calculate the 1st level similarities for all node pairs within the graph as: $S'_{1,2} = 0.512$(neighbour nodes 5, 8, 10), $S'_{3,4} = 0.528$ (neighbour nodes 5, 8, 10), $S'_{5,6} = 0.707$ (neighbour nodes 1, 3, 8, 10), and $S'_{9,10} = 0.633$ (neighbours nodes 1, 3, 5, 8). Let T=0.50, C_1=0.6, C_2=0.4. In Figure 2.4, we connect nodes having 1st level similarities greater than the threshold T by dotted lines. For easier visualization of the vertices within the same record and their associated edges, different colours are used.

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 $S'_{kl} > T$. For nodes 7 and 8, we have the following neighbour pairs (1,2), (3,4), (5,6), (9,10), and we have $S''_{7,8} = \frac{\sqrt{(1\times3)}+\sqrt{(1\times4)}+\sqrt{(1\times2)}+\sqrt{(1\times4)}}{\sqrt{52}} = 0.9909$. Similarly, for nodes 9 and 10, the neighbour pairs are (1,2), (3,4), (5,6), (7,8), and $S''_{9,10} = 0.9902$. The overall similarities, *S* between all attribute values of (R_1, R_6) and (R_3, R_6) are now given by

$$S(R_1, R_6) = \frac{([3-3] + [7-8] + [9-9])}{number\ of\ attributes}$$

$$= \frac{(1 + [C_1 * 0.613 + C_2 * 0.9909] + 1)}{3} = 0.9214 \qquad (7)$$

$$S(R_3, R_6) = 0.9253 \qquad (8)$$

The affinity degree for the possible imputed values is given by the average of the *IS* and S_{ij} measures computed for each possible imputed values. For (Drunk, 3) from R_1, the affinity degree is given by ϑ(Drunk, 3) = (0.9214+1)/2 = 0.9607. For (Drunk, 2) from R_6, ϑ(Drunk, 2) = (0.9253+1)/2 = 0.9626. Finally, the actual imputed value is obtained by random sampling from the list of possible imputed values based on their affinity degrees. For example, since (Drunk, 3) and (Drunk, 2) have affinity degrees of 0.9607 and 0.9626, respectively, their sampling probabilities are 0.4995 and 0.5005, respectively, and both have these probabilities of been chosen as the actual imputed values for the missing values in R_6. Random sampling according to affinity degree ensures that uncertainty and randomness in attribute values are accounted for and helps to reduce systematic bias in the imputed dataset.

2.3.3 DSMI algorithm

The DSMI algorithm is presented below.

DSMI Algorithm:

Step I: Decompose full dataset into complete and missing values sub-datasets: $D_{Full} = D_{Complete} + D_{Miss}$

Step II: Generate a set of decision trees using C4.5 from $D_{Complete}$ where each missing attribute in D_{Miss} produces a tree

Step III: Assign the records in D_{Miss} into leaves of the decision trees and create tables of related records

Step IV: Impute missing values

FOR each table *T* DO

FOR each missing record *R* in *T* DO

Find records in *T* that match with the maximum number of non-missing attribute(s) in the missing record *R*, and let *N* be the number of such records

FOR k = 1 to N determine

O_k = possible imputed value(s) from the k-th matched record

IS_k = *IS* measure computed for O_k

S_k = weighted similarity measure between the k-th matched record and missing record *R*

ϑ_k = affinity degree for O_k

END FOR

Imputed value(s) is obtained by random sampling from the set of possible imputed values $\{O_1 \ldots O_N\}$ based on the sampling probabilities specified by the set of affinity degrees $\{\vartheta_1 \ldots \vartheta_N\}$

END FOR

END FOR

2.4 Experimental results

2.4.1 Datasets

We do experiment on 43 text files data (Large Truck Crash Causation Study File 1 and 2), two datasets on New York's open data portal ("Motor Vehicle Crash-case information: 2011" and "Motor Vehicle Crash- individual information: 2011"), and a dataset of Denver County. The Denver County's dataset includes accidents in the City and County of Denver for the previous five calendar years plus the current year to date (30 June 2014) and has 89194 traffic accident records, in which 1902 records contain missing values. The Large Truck Crash Causation Study Files have different number of attributes and 92871 records, in which 3192 records contain missing values. We removed records having missing values from the datasets so that the datasets only contain complete records. The four datasets are listed in Table 2.11. As can be seen, these datasets contain mostly categorical attributes.

Table 2.11. Datasets details

Dataset	Number of complete records	Categorical attributes	Numerical attributes	As on date
Large Truck Crash Causation Study Files (File 1 and 2) dataset (Truck) [**78**]	89679	A total of 43 text files with different number of attributes in each file, most of the attributes (90%) are categorical		12 February, 2014
Denver County dataset (Denver) [**79**]	87292	13	4	30 June, 2014

Motor vehicle crash- Case information: 2011 dataset (Case) [**76**]	13889	17	1	24 September, 2014
Motor vehicle crash- Individual information: 2011 dataset (Individual) [**76**]	17858	11	3	24 September, 2014

2.4.2 Evaluation criteria

Our proposed imputation accuracy is evaluated using imputation accuracy p and root mean square error (RMSE) [67]. The RMSE is a frequently used measure of the difference between values predicted by a model and the values actually observed. However, RMSE is not appropriate for categorical data as arbitrary value can be assigned to correct/incorrect imputation. As most of the attributes in our traffic accident datasets are categorical, we use p to evaluate categorical imputation accuracy, and only use RMSE for the numerical attributes (if any).

Let n be the total number of missing values and c be the total number of correctly imputed missing value. The accuracy is given by

$$p = \frac{c}{n} \quad (9)$$

The p ranges from 0 to 1, where 1 indicates perfect imputation.

Let A_i be the true value for the i-th missing value, P_i be the imputed value for the i-th missing value, and $e_i = (P_i - A_i)$. The RMSE is given by

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} e_i^2} \quad (10)$$

The lower the RMSE value, the better the imputation.

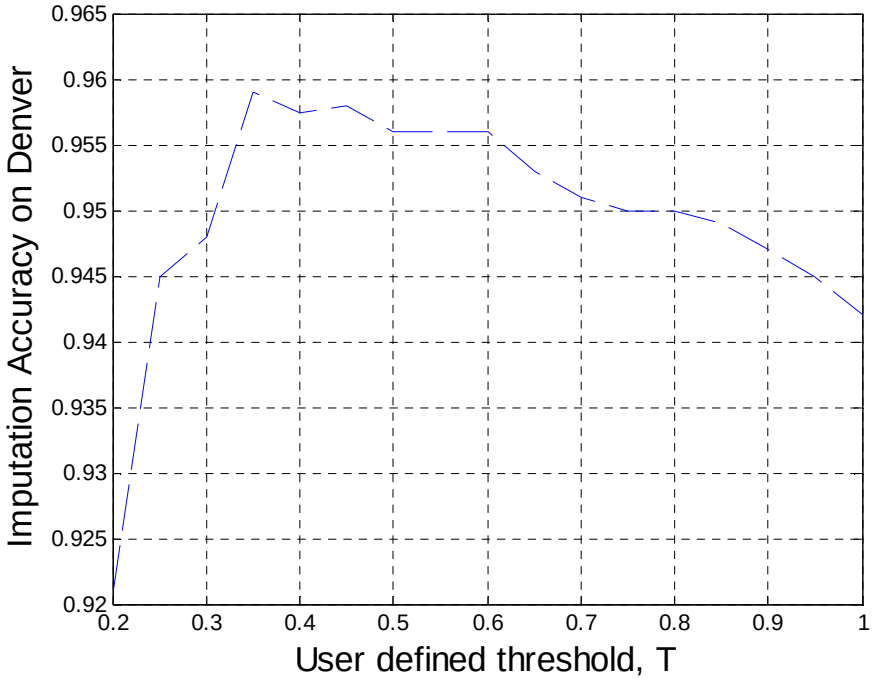
2.4.3 Parameter selection

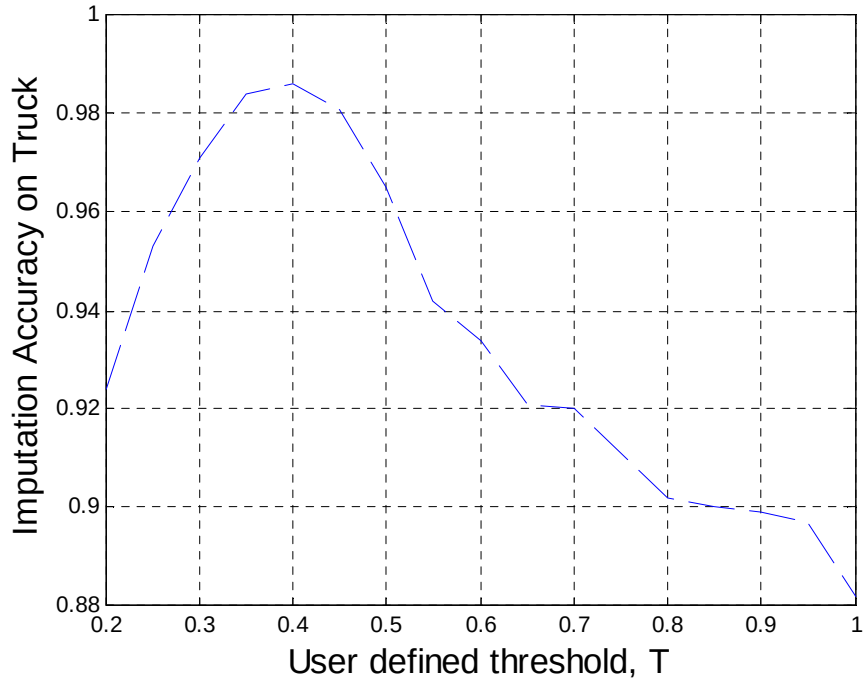
Our proposed algorithm has three parameters that need to be set. These parameters are: similarity threshold T, 1st level similarity weight C_1, and 2nd level similarity weight C_2. Using imputation accuracy p, we analyse the four datasets to select the best threshold parameter T. The results are shown in Figure 2.5. Likewise, we analyse the different values of C_1, C_2 on the four datasets to determine their optimum value. In Table 2.12, we show the effects of the two parameter values on different datasets. Based on this result, we use $C_1 = 0.60$ and $C_2 = 0.40$. Note that these results (Figure 2.5 and Table 2.12) are generated on the four datasets where numerical attributes are excluded.

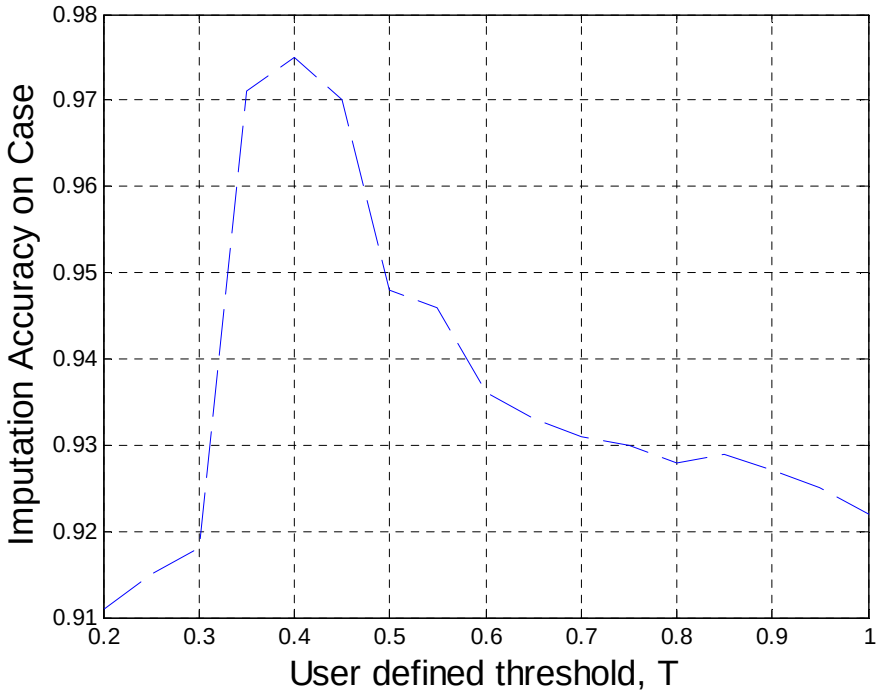
Table 2.12. Similarity measure parameters selection using p

Parameters		Datasets			
C1	C2	Case	Individual	Truck	Denver
1.00	0.00	0.6233	0.5906	0.6845	0.4925
0.95	0.05	0.6240	0.5904	0.6825	0.5221
0.90	0.10	0.7550	0.6864	0.6912	0.5889
0.85	0.15	0.7560	0.6794	0.7180	0.6851
0.80	0.20	0.7665	0.7776	0.7762	0.6834
0.75	0.25	0.8275	0.7855	0.7941	0.7814
0.70	0.30	0.8899	0.8514	0.8701	0.8210
0.65	**0.35**	**0.9612**	**0.9599**	**0.9541**	**0.9598**
0.60	**0.40**	**0.9782**	**0.9644**	**0.9810**	**0.9566**
0.55	**0.45**	**0.9538**	**0.9644**	**0.9564**	**0.9245**
0.50	0.50	0.8657	0.9648	0.8678	0.8698
0.45	0.55	0.8688	0.9104	0.8698	0.7714
0.40	0.60	0.8004	0.9024	0.8010	0.7802
0.35	0.65	0.7915	0.8846	0.8120	0.6203

0.30	0.70	0.7819	0.8556	0.7922	0.5892
0.25	0.75	0.6910	0.7344	0.7100	0.5099
0.20	0.80	0.6926	0.7332	0.7200	0.4345
0.15	0.85	0.6010	0.7113	0.7010	0.4135
0.10	0.90	0.6199	0.7209	0.6100	0.4214
0.05	0.95	0.6146	0.6324	0.6204	0.4365
0.00	1.00	0.6098	0.6216	0.6202	0.4404




Imputation Accuracy on Truck
User defined threshold, T
1
0.98
0.96
0.94
0.92
0.9
0.88
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


Imputation Accuracy on Case
User defined threshold, T
0.98
0.97
0.96
0.95
0.94
0.93
0.92
0.91
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

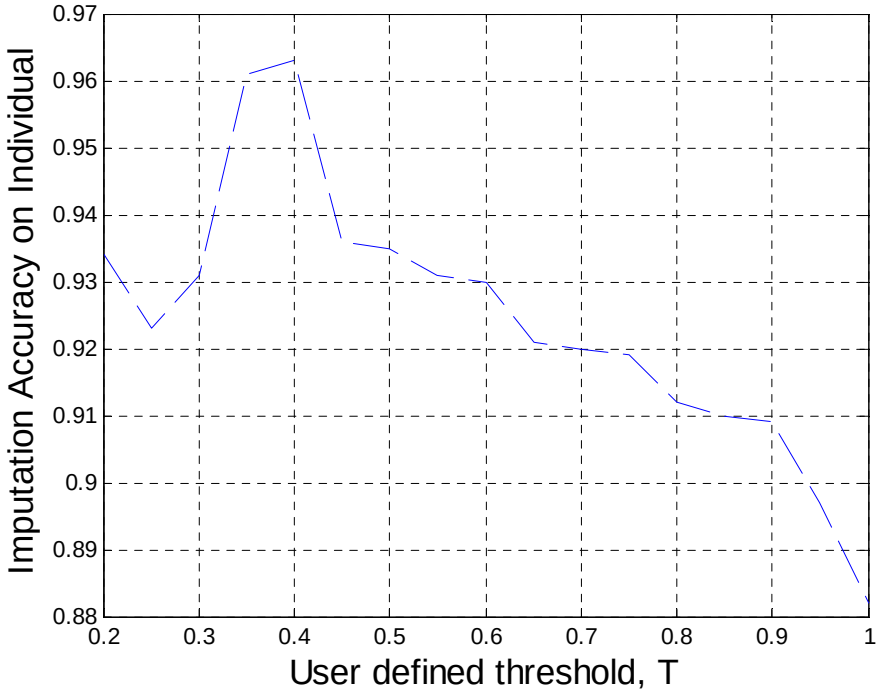


Figure 2.5: Threshold parameter T on four datasets

2.4.4 Missing value simulation

We use four types of missing patterns: simple, medium, complex, and blended [3, 17]. In simple pattern, a record can have at most one missing value. In medium pattern, a record can have missing values for up to 50 % of the total number of attributes. In a complex pattern, a record can have missing values for up to 80 % of the total number of attributes. A blended pattern contains 25% records having missing values with simple pattern, 50% with medium pattern and 25% with complex pattern. For each missing pattern, we use four missing ratios: 2%, 4%, 8% and 10%. We use two types of missing models, namely overall and uniformly distributed (UD). In the UD missing model, each attribute has equal number of missing values. However, in the overall model, missing values are not equally distributed among the attributes and in the worst case all missing values can belong to a single attribute [3, 83].

In our experiments, we artificially create missing values in the dataset by using 4 missing patterns, namely simple, medium, complex and blended, 4 missing ratios i.e. 2%, 4%, 8% and 10%, and 2 missing models, namely overall and uniformly distributed (UD). We have altogether 32 missing combinations (4 missing ratios × 4 missing patterns × 2 missing models). For each combination, we generate 20 datasets i.e. in total we create 640 datasets (32 combinations × 20 datasets per combination) with missing values for each real dataset, as shown in Table 2.13.

Table 2.13. Missing value simulation

Missing patterns	Number of attributes having missing values	Missing ratios	Missing model	Number of datasets for each pattern
Simple	1	2%, 4%, 8% and 10%	Overall and Uniformly distributed	20
Medium	Up to 50%			
Complex	Up to 80%			
Blended	Simple-25%, Medium-50% and Complex-25%			

2.4.5 Results of categorical missing values imputation

For categorical missing value imputation, we compare our proposed algorithm DSMI with five imputation methods FEMI [32], FIMUS [29], (Grey + CPAANN) [24], DMI [31], and KNNI [20]. We present the imputation accuracy of DSMI, FEMI, (Grey + CPAANN), FIMUS, DMI, and KNNI on the "Motor vehicle crash case information: 2011 dataset", "Truck crash causation dataset", "Denver County dataset", and "Motor vehicle crash individual information: 2011 dataset" in Table 2.14, Table 2.15, Table 2.16, and Table 2.17 respectively. Only categorical attributes are considered here. From each dataset, we generate 32 combinations of four missing ratios, two missing models, and four missing patterns. Each value in these tables is the average of 20 imputation runs carried out on 20 datasets. In these two tables, bold values mark the best imputation result compare with other imputation methods. From these tables, it can be

seen that our DSMI imputation method performs significantly better than FEMI, FIMUS, Grey + CPAANN, DMI and KNNI methods.

Table 2.14. Performance on Motor Vehicle Crash case information: 2011 dataset

Missing combination			Accuracy (p)					
			DSMI	FEMI	FIMUS	Grey+CPAANN	DMI	KNNI
2%	Overall	Simple	**0.9782**	0.8301	0.8130	0.8090	0.7110	0.6310
		Medium	**0.9671**	0.8300	0.8160	0.8088	0.7110	0.6210
		Complex	**0.9634**	0.8267	0.8020	0.8082	0.7190	0.6270
		Blended	**0.9626**	0.8260	0.8020	0.8032	0.7190	0.6260
	UD	Simple	**0.9682**	0.8300	0.8220	0.8031	0.7110	0.6390
		Medium	**0.9662**	0.8289	0.8030	0.8011	0.7090	0.6280
		Complex	**0.9622**	0.8245	0.8010	0.8001	0.7070	0.6220
		Blended	**0.9632**	0.8240	0.8010	0.8000	0.7060	0.6220
4%	Overall	Simple	**0.9582**	0.8262	0.8100	0.8008	0.7095	0.6290
		Medium	**0.9580**	0.8261	0.8160	0.8002	0.7085	0.6270
		Complex	**0.9530**	0.8134	0.8060	0.8001	0.7090	0.6010
		Blended	**0.9520**	0.8104	0.8250	0.8000	0.7070	0.6220
	UD	Simple	**0.9566**	0.8200	0.8040	0.7909	0.7010	0.6290
		Medium	**0.9560**	0.8165	0.8080	0.7908	0.7030	0.6260
		Complex	**0.9540**	0.8123	0.8090	0.7902	0.7080	0.6020
		Blended	**0.9540**	0.8103	0.8070	0.7901	0.7060	0.6210
8%	Overall	Simple	**0.9562**	0.8101	0.8080	0.7903	0.6980	0.6140
		Medium	**0.9560**	0.8100	0.8080	0.7902	0.6960	0.6150
		Complex	**0.9480**	0.8023	0.8040	0.7900	0.6930	0.6144
		Blended	**0.9500**	0.7989	0.8020	0.7825	0.6910	0.6160

	UD	Simple	**0.9542**	0.8078	0.8060	0.7820	0.6902	0.6140
		Medium	**0.9510**	0.8070	0.8070	0.7808	0.6900	0.6040
		Complex	**0.9470**	0.8060	0.8040	0.7730	0.6823	0.6050
		Blended	**0.9480**	0.8045	0.7920	0.7705	0.6820	0.6140
10%	Overall	Simple	**0.9500**	0.7967	0.7810	0.7800	0.6805	0.6003
		Medium	**0.9480**	0.7960	0.7910	0.7701	0.6801	0.6000
		Complex	**0.9450**	0.7923	0.7840	0.7645	0.6789	0.5956
		Blended	**0.9470**	0.7840	0.7835	0.7640	0.6723	0.5923
	UD	Simple	**0.9510**	0.7831	0.7810	0.7605	0.6720	0.5980
		Medium	**0.9460**	0.7810	0.7800	0.7600	0.6703	0.5960
		Complex	**0.9490**	0.7800	0.7840	0.7565	0.6612	0.5879
		Blended	**0.9470**	0.7800	0.7845	0.7510	0.6610	0.5870

Table 2.15. Performance on Large Truck Crash Causation dataset

			Accuracy (p)					
Missing combination			DSMI	FEMI	FIMUS	Grey+CPAANN	DMI	KNNI
2%	Overall	Simple	**0.9810**	0.9233	0.9100	0.9089	0.7980	0.7190
		Medium	**0.9792**	0.9230	0.9110	0.9088	0.7860	0.7100
		Complex	**0.9762**	0.9210	0.9150	0.9070	0.7950	0.7070
		Blended	**0.9742**	0.9201	0.9160	0.9050	0.7940	0.7050
	UD	Simple	**0.9778**	0.9189	0.9090	0.9007	0.7970	0.7010
		Medium	**0.9748**	0.9180	0.9080	0.9001	0.7980	0.7000
		Complex	**0.9726**	0.9067	0.8960	0.8823	0.7900	0.6987
		Blended	**0.9720**	0.9023	0.8960	0.8803	0.7890	0.6980

4%	Overall	Simple	**0.9752**	0.9001	0.8860	0.8780	0.7920	0.6930
		Medium	**0.9736**	0.8980	0.8860	0.8778	0.7730	0.6910
		Complex	**0.9714**	0.8930	0.8110	0.8770	0.7630	0.6815
		Blended	**0.9726**	0.8904	0.8190	0.8777	0.7620	0.6844
	UD	Simple	**0.9730**	0.8803	0.8130	0.8585	0.7530	0.6830
		Medium	**0.9730**	0.8745	0.8140	0.8580	0.7510	0.6770
		Complex	**0.9702**	0.8503	0.7690	0.85005	0.7490	0.6730
		Blended	**0.9704**	0.8500	0.7680	0.8401	0.7480	0.6720
8%	Overall	Simple	**0.9670**	0.8456	0.8140	0.8301	0.7490	0.6710
		Medium	**0.9660**	0.8405	0.8160	0.8300	0.7460	0.6760
		Complex	**0.9640**	0.8400	0.8080	0.8204	0.7330	0.6610
		Blended	**0.9610**	0.8378	0.8150	0.8190	0.7320	0.6690
	UD	Simple	**0.9620**	0.8320	0.8020	0.8109	0.7280	0.6600
		Medium	**0.9620**	0.8301	0.8140	0.8100	0.7250	0.6580
		Complex	**0.9568**	0.8300	0.8030	0.8056	0.7210	0.6510
		Blended	**0.9584**	0.8209	0.8040	0.8050	0.7200	0.6420
10%	Overall	Simple	**0.9554**	0.7989	0.8040	0.8006	0.7140	0.6370
		Medium	**0.9524**	0.7980	0.8030	0.8001	0.7130	0.6360
		Complex	**0.9510**	0.7823	0.7900	0.7978	0.7100	0.6300
		Blended	**0.9522**	0.7820	0.7960	0.8000	0.7100	0.6280
	UD	Simple	**0.9532**	0.7800	0.7910	0.7989	0.7020	0.6270
		Medium	**0.9514**	0.7749	0.7800	0.7856	0.7010	0.6150
		Complex	**0.9496**	0.7740	0.7940	0.7700	0.6980	0.6140
		Blended	**0.9460**	0.7706	0.7820	0.7673	0.6940	0.6010

Table 2.16. Performance on Denver County dataset

Missing combination			Accuracy (p)					
			DSMI	FEMI	FIMUS	Grey+CPAANN	DMI	KNNI
2%	Overall	Simple	**0.9671**	0.8267	0.8009	0.8045	0.7145	0.7267
		Medium	**0.9649**	0.8261	0.8004	0.8041	0.7135	0.7261
		Complex	**0.9625**	0.8250	0.8001	0.8038	0.7130	0.7260
		Blended	**0.9617**	0.8187	0.8000	0.8031	0.7009	0.7255
	UD	Simple	**0.9621**	0.8170	0.7923	0.8003	0.7007	0.7251
		Medium	**0.9620**	0.8104	0.7920	0.8002	0.7003	0.7250
		Complex	**0.9519**	0.8101	0.7919	0.7980	0.7002	0.7105
		Blended	**0.9511**	0.8100	0.7910	0.7935	0.7000	0.7100
4%	Overall	Simple	**0.9598**	0.8262	0.8104	0.8008	0.6104	0.7114
		Medium	**0.9580**	0.8261	0.7934	0.8002	0.6104	0.7114
		Complex	**0.9575**	0.8134	0.7932	0.8001	0.6004	0.6124
		Blended	**0.9570**	0.8104	0.7904	0.8000	0.6005	0.6114
	UD	Simple	**0.9562**	0.8200	0.7804	0.7909	0.6105	0.6324
		Medium	**0.9550**	0.8165	0.7703	0.7908	0.6405	0.6225
		Complex	**0.9548**	0.8123	0.7604	0.7902	0.6105	0.6125
		Blended	**0.9545**	0.8103	0.7103	0.7901	0.6105	0.5126
8%	Overall	Simple	**0.9531**	0.8009	0.7831	0.7819	0.6987	0.6645
		Medium	**0.9522**	0.8011	0.7820	0.7816	0.6976	0.6621
		Complex	**0.9519**	0.8002	0.7810	0.7811	0.6902	0.6610
		Blended	**0.9447**	0.8000	0.7801	0.7810	0.6900	0.6645
	UD	Simple	**0.9511**	0.7989	0.7800	0.7801	0.6873	0.6531
		Medium	**0.9501**	0.7981	0.7767	0.7800	0.6833	0.6511
		Complex	**0.9424**	0.7979	0.7734	0.7787	0.6821	0.6501
		Blended	**0.9420**	0.7971	0.7732	0.7776	0.6819	0.6500

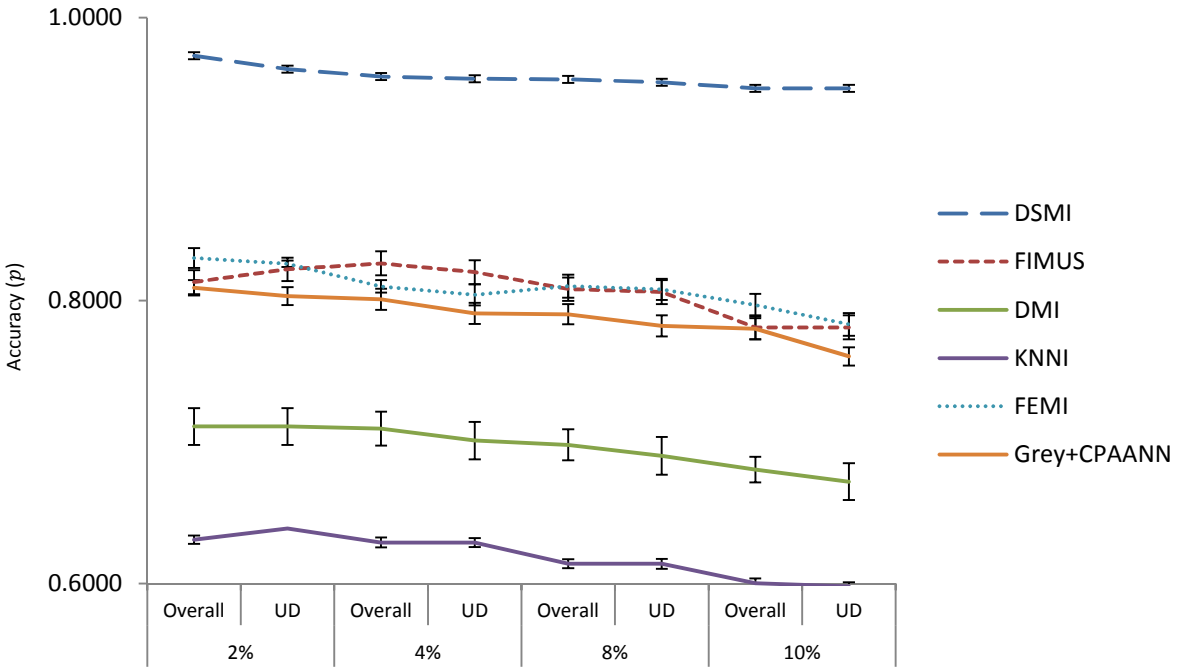
10%	Overall	Simple	**0.9494**	0.7838	0.7809	0.7754	0.6845	0.6332
		Medium	**0.9465**	0.7831	0.7802	0.7743	0.6765	0.6311
		Complex	**0.9461**	0.7830	0.7667	0.7523	0.6671	0.6307
		Blended	**0.9454**	0.7823	0.7613	0.7521	0.6617	0.6302
	UD	Simple	**0.9405**	0.7821	0.7606	0.7465	0.6610	0.6300
		Medium	**0.9403**	0.7811	0.7604	0.7609	0.6607	0.6209
		Complex	**0.9401**	0.7808	0.7601	0.7402	0.6603	0.6205
		Blended	**0.9400**	0.7765	0.7600	0.7389	0.6601	0.6200

Table 2.17. Performance on Motor Vehicle Crash- individual information: 2011 dataset

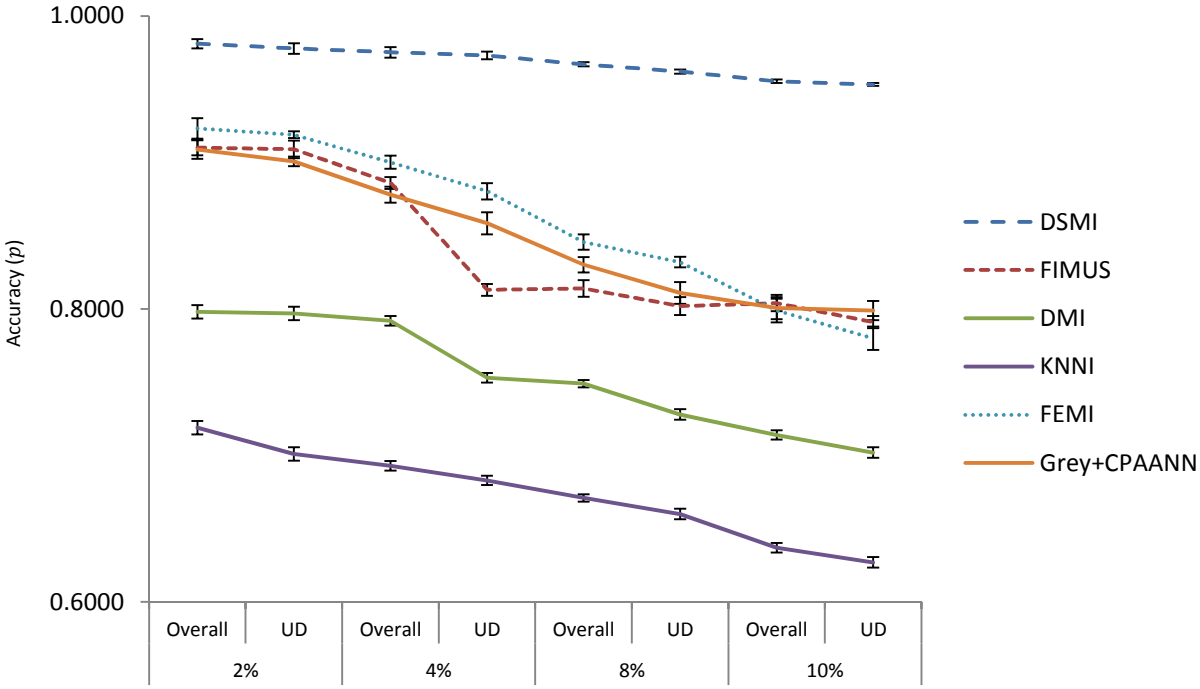
Missing combination			Accuracy (p)					
			DSMI	FEMI	FIMUS	Grey+CPAANN	DMI	KNNI
2%	Overall	Simple	**0.9720**	0.8610	0.8435	0.8417	0.7609	0.7308
		Medium	**0.9710**	0.8603	0.8430	0.8410	0.7600	0.7301
		Complex	**0.9654**	0.8545	0.8403	0.8405	0.7521	0.7274
		Blended	**0.9650**	0.8522	0.8400	0.8501	0.7511	0.7217
	UD	Simple	**0.9701**	0.8511	0.8387	0.8433	0.7506	0.7301
		Medium	**0.9680**	0.8510	0.8376	0.8423	0.7502	0.7234
		Complex	**0.9665**	0.8504	0.8325	0.8413	0.7456	0.7205
		Blended	**0.9650**	0.8503	0.8321	0.8410	0.7450	0.7201
4%	Overall	Simple	**0.9651**	0.8500	0.8302	0.8405	0.7522	0.7054
		Medium	**0.9612**	0.8467	0.8301	0.8403	0.7510	0.7034
		Complex	**0.9605**	0.8432	0.8300	0.8400	0.7423	0.7004
		Blended	**0.9601**	0.8413	0.8300	0.8398	0.7410	0.7001
	UD	Simple	**0.9603**	0.8410	0.8298	0.8378	0.7445	0.6704
		Medium	**0.9600**	0.8407	0.8290	0.8370	0.7424	0.6700

		Complex	**0.9587**	0.8403	0.8265	0.8319	0.7414	0.6623
		Blended	**0.9672**	0.8402	0.8245	0.8310	0.7410	0.6603
8%	Overall	Simple	**0.9634**	0.8380	0.8226	0.8311	0.7401	0.6601
		Medium	**0.9632**	0.8365	0.8217	0.8310	0.7400	0.6600
		Complex	**0.9622**	0.8356	0.8206	0.8307	0.7372	0.6453
		Blended	**0.9611**	0.8316	0.8202	0.8274	0.7327	0.6413
	UD	Simple	**0.9605**	0.8311	0.8201	0.8223	0.7317	0.6512
		Medium	**0.9604**	0.8310	0.8201	0.8221	0.7302	0.6500
		Complex	**0.9602**	0.8304	0.8167	0.8204	0.7256	0.6434
		Blended	**0.9601**	0.8302	0.8157	0.8201	0.7203	0.6423
10%	Overall	Simple	**0.9548**	0.8156	0.8108	0.8176	0.7204	0.6406
		Medium	**0.9540**	0.8154	0.8106	0.8174	0.7203	0.6402
		Complex	**0.9535**	0.8107	0.8101	0.8056	0.7187	0.6345
		Blended	**0.9505**	0.8100	0.8073	0.8055	0.7145	0.6314
	UD	Simple	**0.9502**	0.8098	0.8009	0.8001	0.7200	0.6321
		Medium	**0.9501**	0.8030	0.8010	0.7987	0.7134	0.6309
		Complex	**0.9445**	0.8000	0.7945	0.7927	0.7032	0.6230
		Blended	**0.9441**	0.7976	0.7910	0.7821	0.7003	0.6210

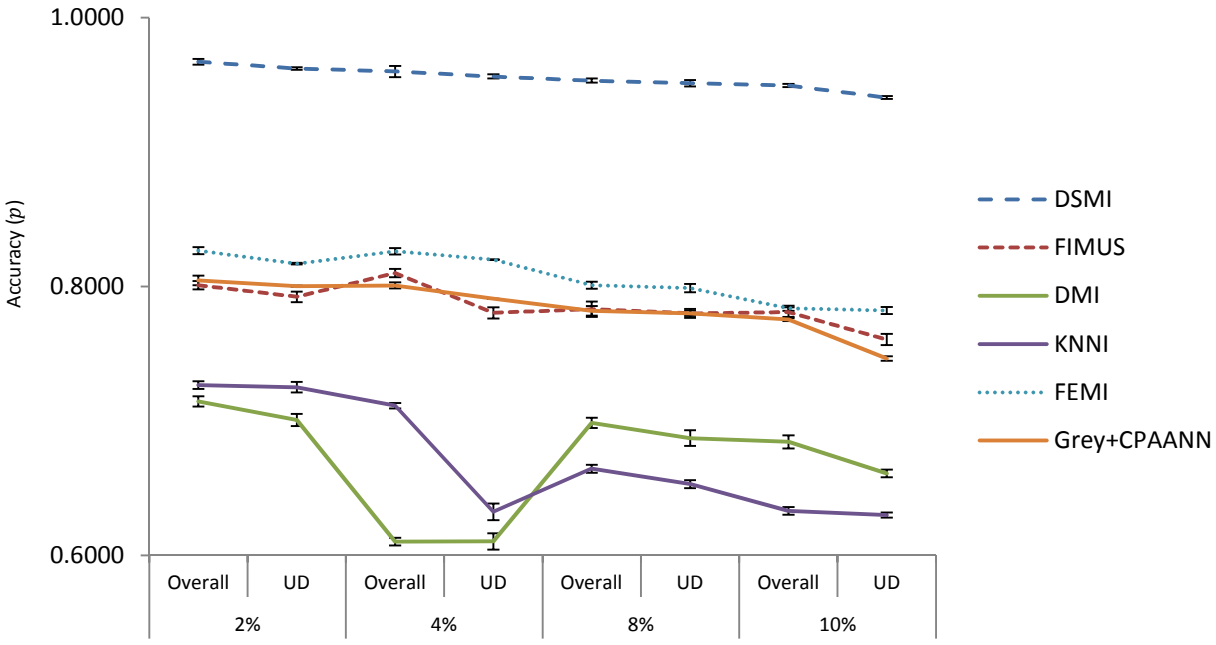
We present the result for the four traffic accident datasets on all 8 combinations of missing ratios and missing models with the simple missing pattern in Figure 2.6 and medium missing pattern in Figure 2.7. Here, performance is evaluated with 95% confidence levels. Figure 2.8 presents the result for the four traffic accident datasets on 4% missing ratio and 8 missing combinations of missing patterns and missing models with 95% confidence levels.



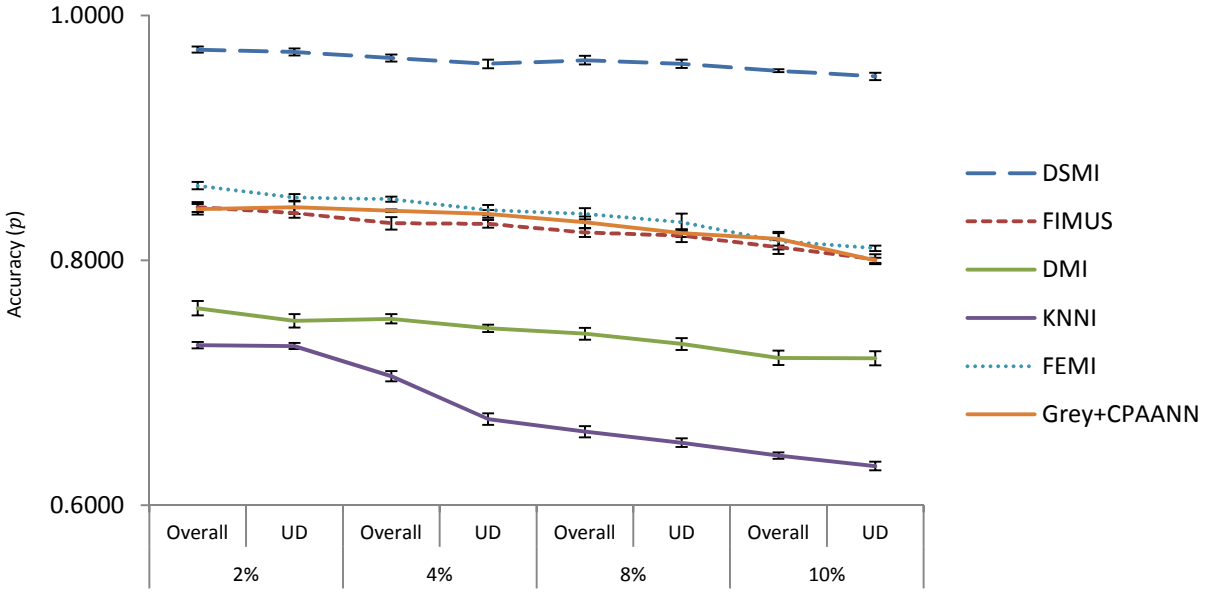
(a) Case dataset



(b) Truck dataset



(c) Denver County dataset



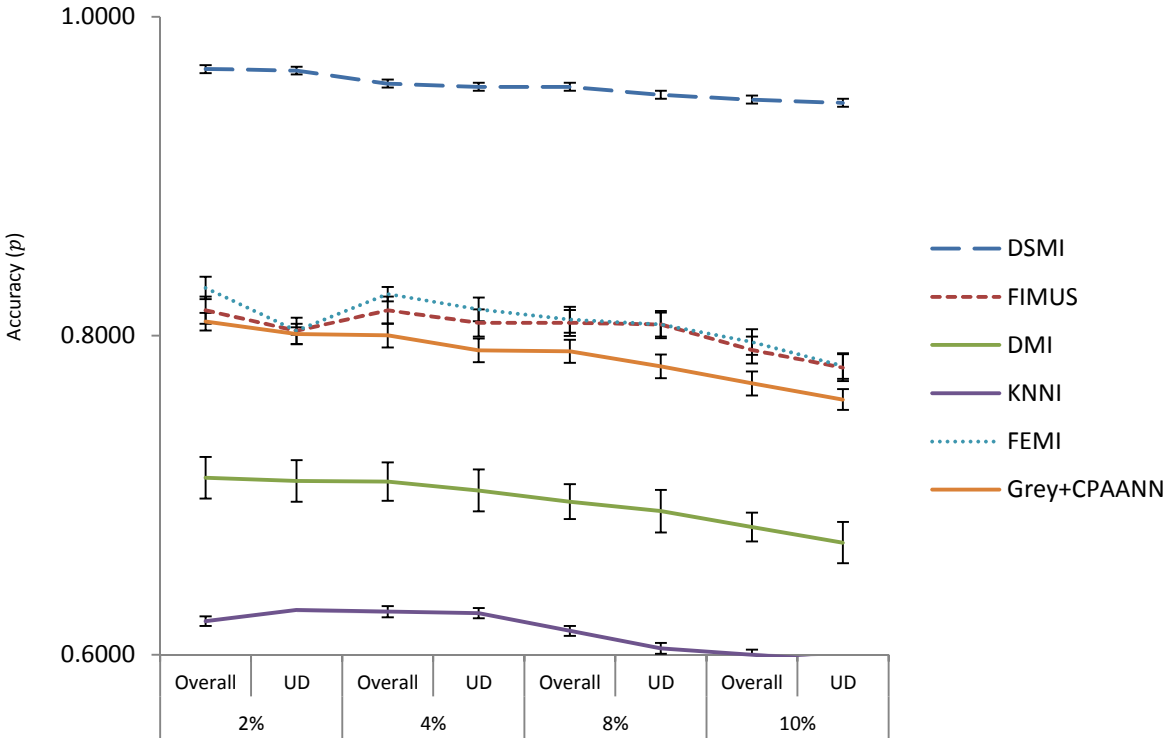
(d) Individual dataset

Figure 2.6. Aggregated Performance (p) on four datasets in terms of "simple" missing pattern with

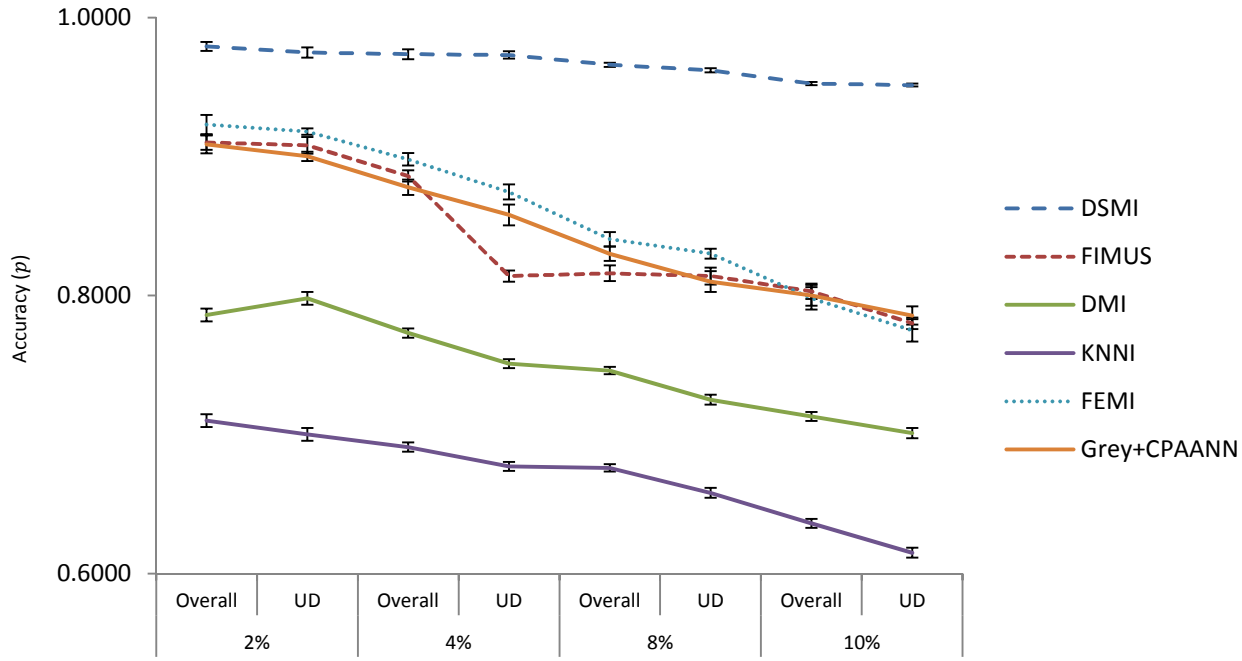
confidence level 95 percent

Figure 2.7 shows the result for the four traffic accident datasets on all 8 combinations of missing ratios and missing models with the medium missing pattern. Here, performance is evaluated with 95% confidence levels.

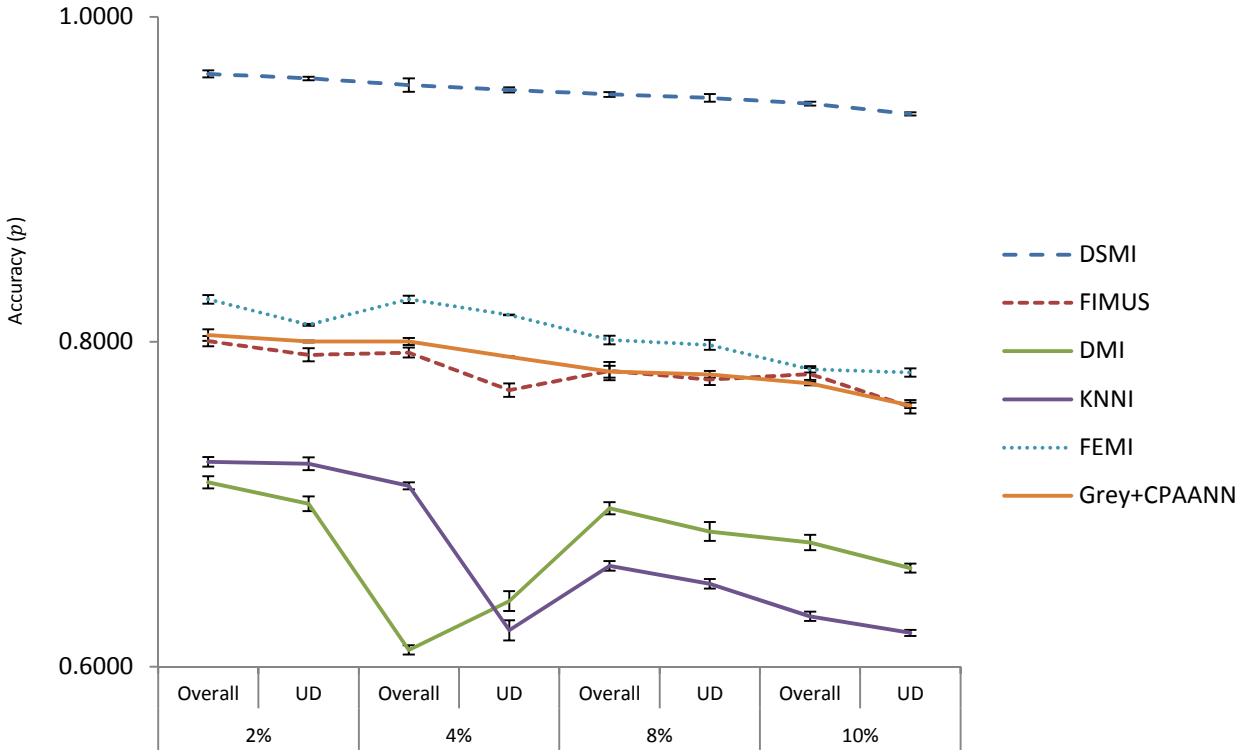
Figure 2.8 presents the result for the four traffic accident datasets on 4% missing ratio and 8 missing combinations of missing patterns and missing models with 95% confidence levels.



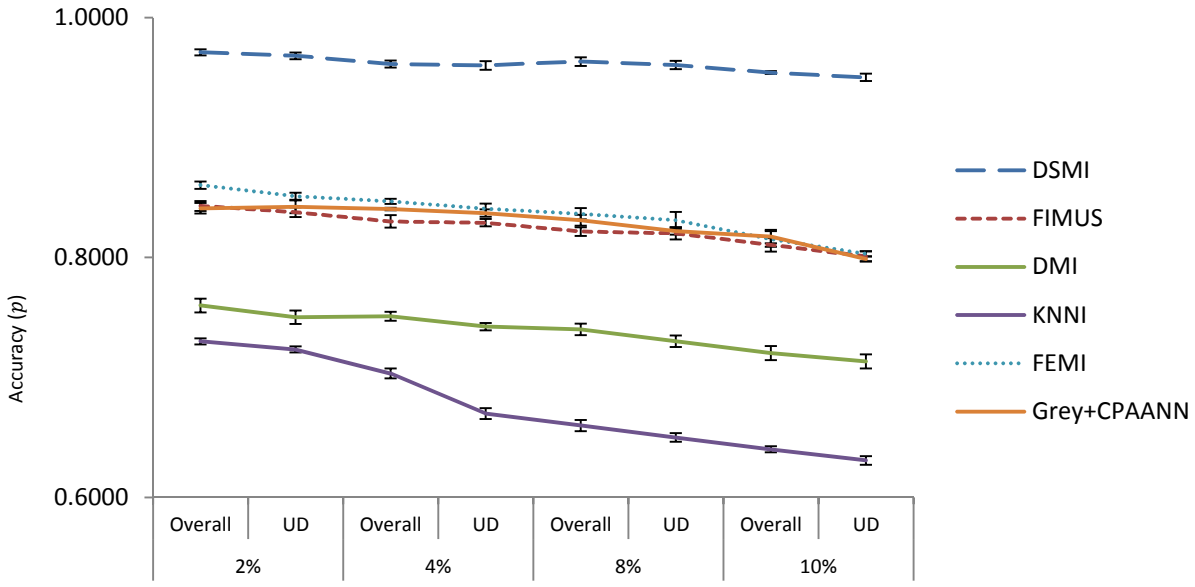
(a) Case dataset



(b) Truck dataset

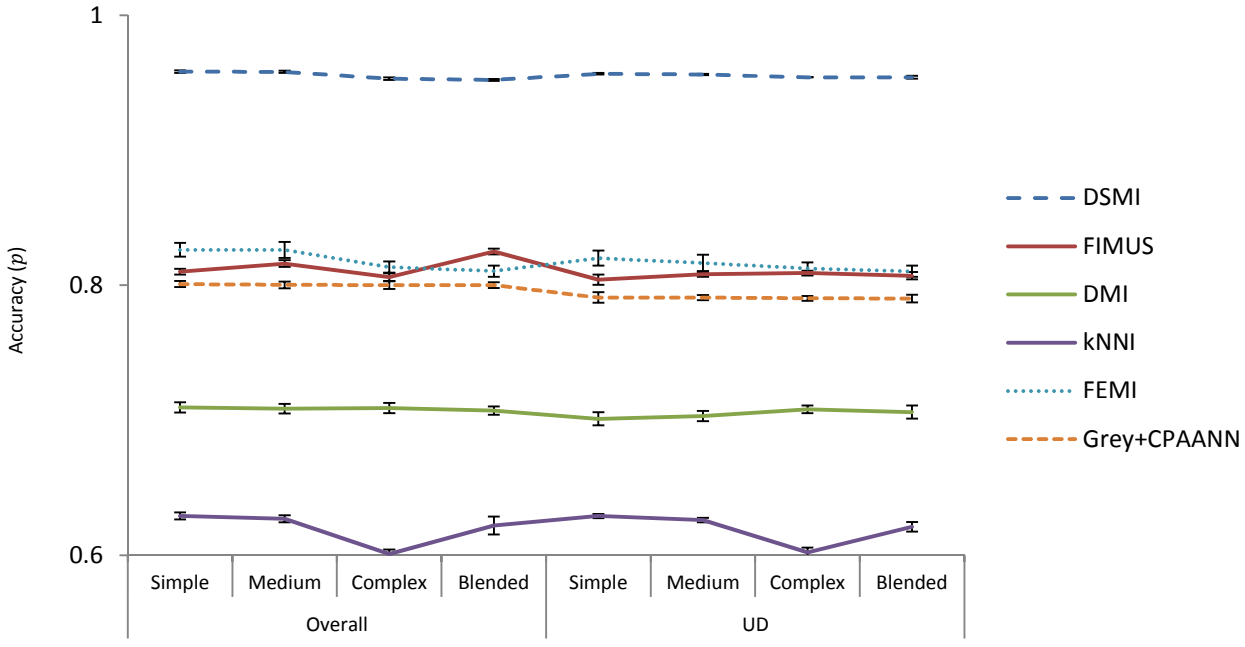


(c) Denver County dataset

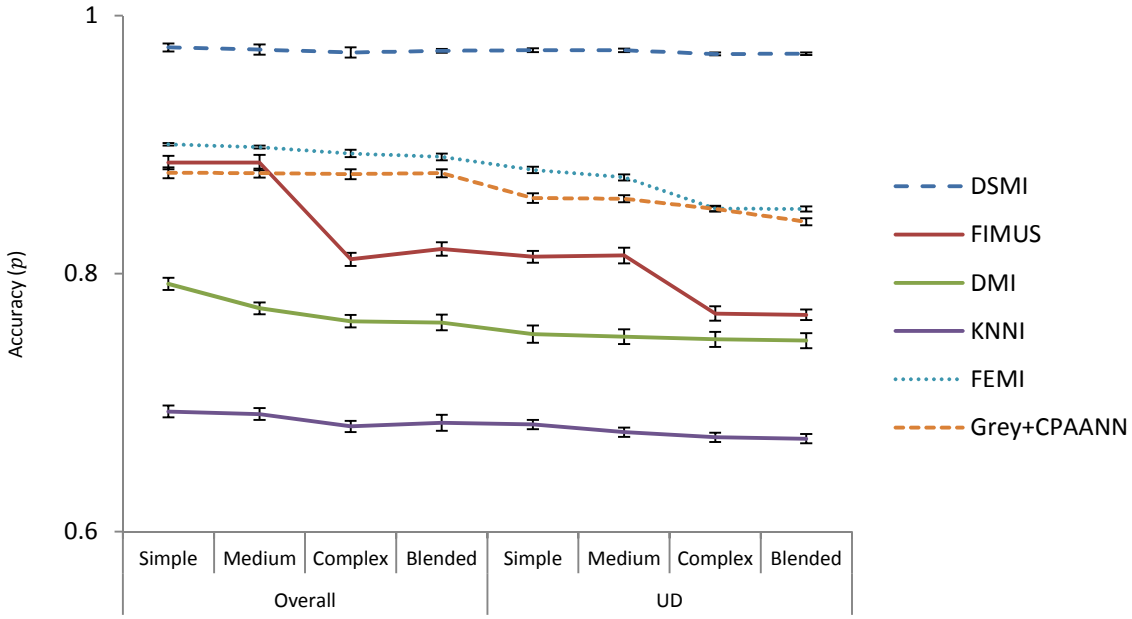


(d) Individual dataset

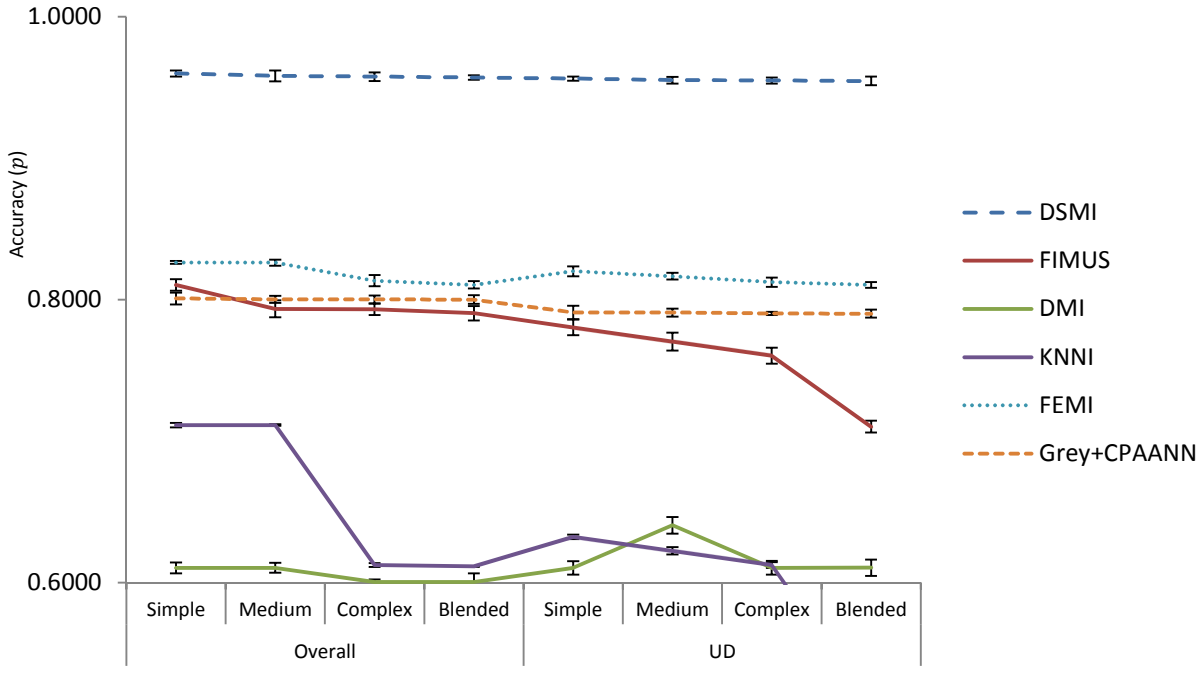
Figure 2.7. Aggregated Performance (p) on four datasets in terms of "medium" missing pattern with confidence level 95 percent



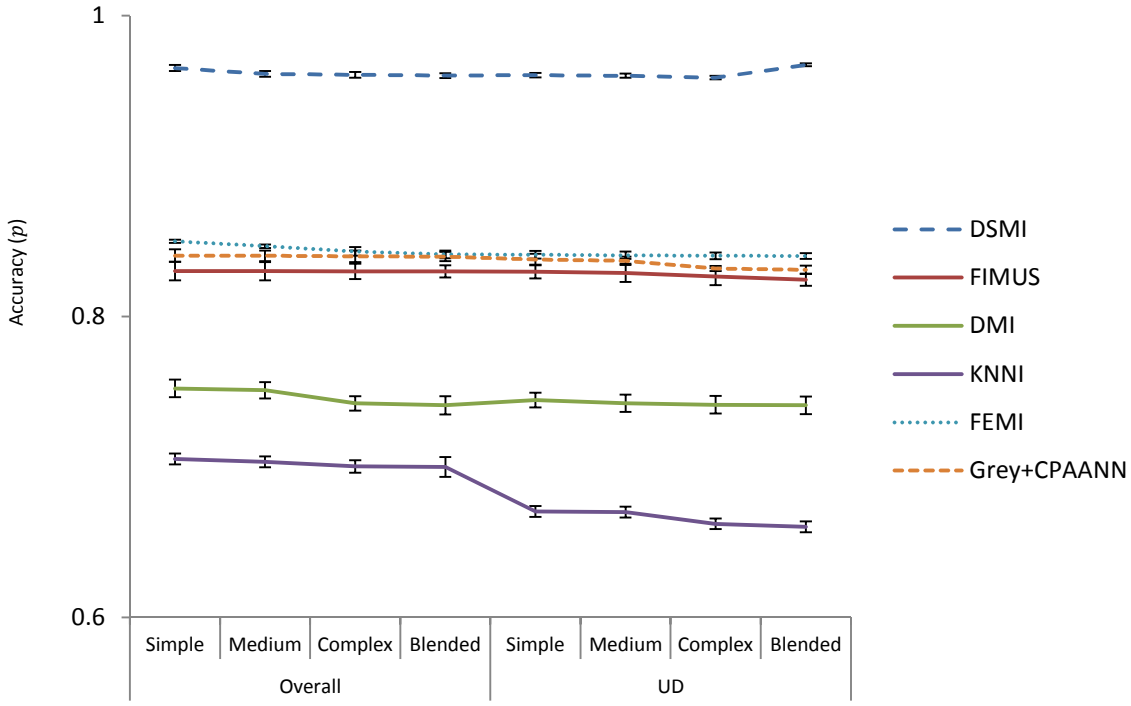
(a) Case dataset



(b) Truck dataset



(c) Denver County dataset



(d) Individual dataset

Figure 2.8. Aggregated Performance (p) on four datasets in terms of "4%" missing ratio with confidence level 95 percent

Figure 2.9 presents the average performance indicators (for 32 missing combinations) for each dataset. It is clearly shown that DSMI performs better than the five existing algorithms. From this graph, it can be seen that our DSMI imputation method performs significantly better than FEMI, FIMUS, Grey + CPAANN, DMI and KNNI methods.

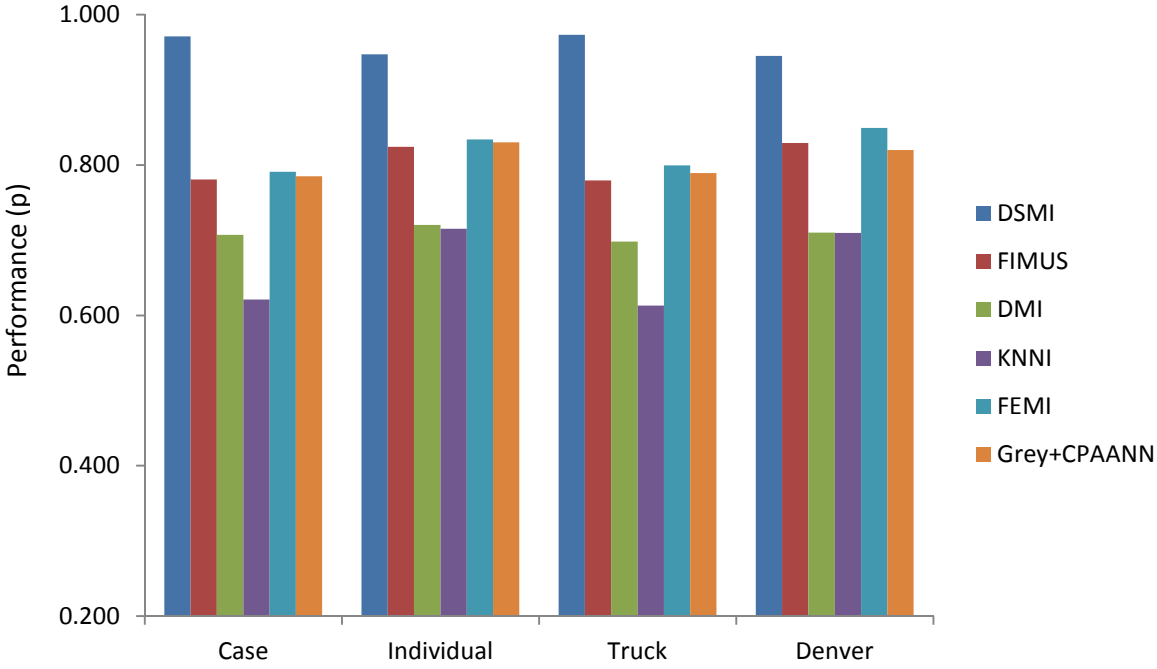


Figure 2.9. Performance (p) comparison on four datasets

2.4.6 Results of numerical missing values imputation

For numerical missing value imputation, we present the RMSE value of DSMI, FEMI [32], FIMUS [29], (Grey + CPAANN) [24], DMI [31], EMI [33] and KNNI [20] on the dataset of "Motor vehicle crash individual information: 2011" in Table 2.18, considering the imputation accuracy for the three numerical attributes of this dataset. We aggregate the results based on four missing ratios, two missing models, and four missing patterns. In the table, bold values mark the best imputation result compare with other imputation methods. From the table, it can be seen that our DSMI imputation method performs significantly better than FEMI, FIMUS, (Grey + CPAANN), DMI, EMI and KNNI methods.

Table 2.18. Performance (RMSE) on Motor Vehicle Crash individual information dataset

Missing combination			RMSE						
			DSMI	FEMI	FIMUS	Grey+CPAANN	DMI	EMI	KNNI
2%	Overall	Simple	**0.0930**	0.1090	0.1120	0.1102	0.1190	0.1580	0.1320
		Medium	**0.0930**	0.1091	0.1130	0.1102	0.1200	0.1590	0.1340
		Complex	**0.0970**	0.1095	0.1135	0.1105	0.1210	0.1600	0.1390
		Blended	**0.0960**	0.1099	0.1140	0.1109	0.1220	0.1610	0.1380
	UD	Simple	**0.1041**	0.1105	0.1120	0.1110	0.1220	0.1590	0.1360
		Medium	**0.1042**	0.1106	0.1140	0.1121	0.1230	0.1600	0.1340
		Complex	**0.1060**	0.1110	0.1150	0.1126	0.1230	0.1610	0.1390
		Blended	**0.1050**	0.1112	0.1150	0.1128	0.1240	0.1610	0.1400
4%	Overall	Simple	**0.0938**	0.1140	0.1130	0.1131	0.1210	0.1640	0.1380
		Medium	**0.0940**	0.1142	0.1130	0.1132	0.1240	0.1660	0.1400
		Complex	**0.1050**	0.1144	0.1145	0.1141	0.1260	0.1670	0.1460
		Blended	**0.1054**	0.1148	0.1150	0.1151	0.1280	0.1660	0.1480
	UD	Simple	**0.1040**	0.1150	0.1140	0.1161	0.1230	0.1680	0.1400
		Medium	**0.1041**	0.1152	0.1160	0.1165	0.1230	0.1680	0.1410
		Complex	**0.1051**	0.1180	0.1170	0.1171	0.1290	0.1710	0.1450
		Blended	**0.1051**	0.1182	0.1175	0.1175	0.1280	0.1720	0.1490
8%	Overall	Simple	**0.1041**	0.1201	0.1145	0.1181	0.1310	0.1730	0.1390
		Medium	**0.1042**	0.1202	0.1160	0.1186	0.1310	0.1740	0.1420
		Complex	**0.1058**	0.1210	0.1190	0.1185	0.1330	0.1810	0.1490
		Blended	**0.1059**	0.1212	0.1198	0.1196	0.1350	0.1820	0.1500
	UD	Simple	**0.1041**	0.1230	0.1185	0.1201	0.1330	0.1800	0.1400
		Medium	**0.1043**	0.1235	0.1195	0.1205	0.1320	0.1840	0.1440
		Complex	**0.1060**	0.1241	0.1200	0.1210	0.1380	0.1860	0.1520

		Blended	**0.1061**	0.1255	0.1210	0.1246	0.1360	0.1880	0.1560
10%	Overall	Simple	**0.1043**	0.1261	0.1170	0.1243	0.1330	0.1870	0.1470
		Medium	**0.1042**	0.1265	0.1180	0.1265	0.1360	0.1880	0.1480
		Complex	**0.1059**	0.1251	0.1230	0.1266	0.1400	0.1910	0.1610
		Blended	**0.1060**	0.1271	0.1240	0.1271	0.1410	0.1900	0.1640
	UD	Simple	**0.1045**	0.1280	0.1190	0.1281	0.1390	0.1890	0.1490
		Medium	**0.1047**	0.1286	0.1200	0.1286	0.1400	0.1900	0.1520
		Complex	**0.1062**	0.1288	0.1250	0.1288	0.1470	0.1930	0.1690
		Blended	**0.1064**	0.1290	0.1260	0.1291	0.1480	0.1960	0.1720

We compare the RMSE value for the numerical missing values in Figure 2.10. We present the overall average (for 32 missing combinations) of RMSE for DSMI, FEMI, (Grey + CPAANN), FIMUS, DMI, KNNI and EMI on all four datasets. This result is generated on four datasets where categorical attributes are excluded.

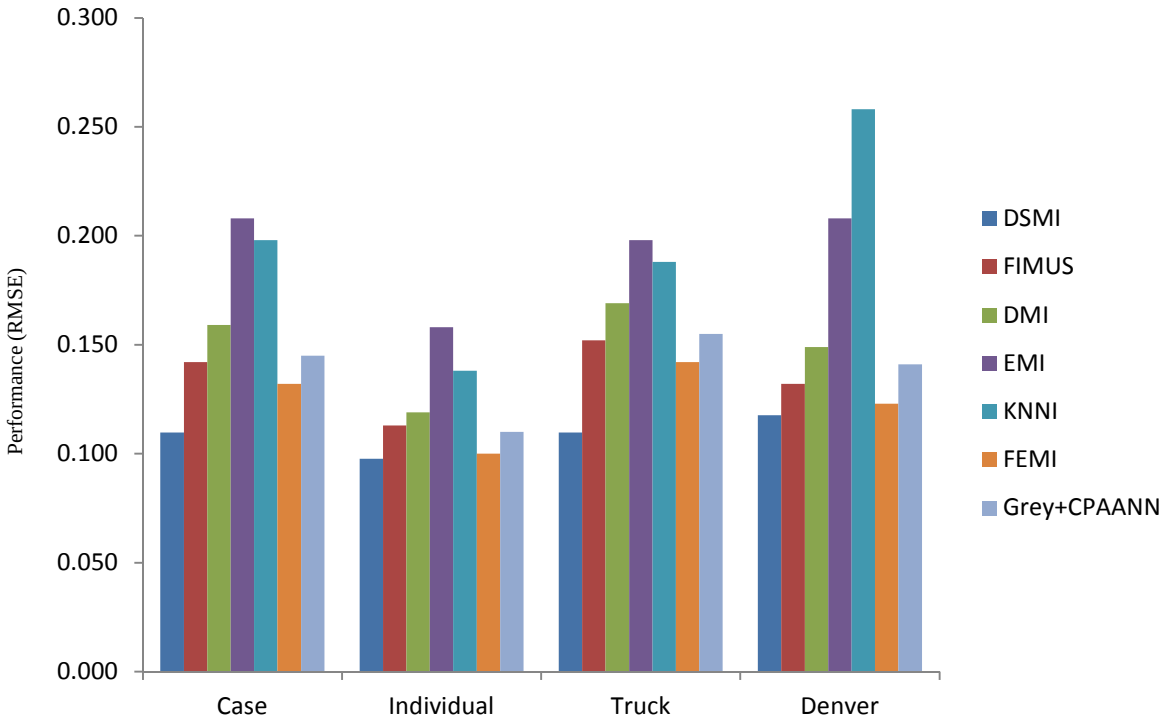


Figure 2.10. Performance comparison for numerical imputation on four datasets using RMSE

2.4.7 Execution time comparison

In Table 2.19, we present the average computation time in seconds for 640 datasets (32 combinations × 20 datasets per combination) for each natural dataset described in Table 2.11. The configuration of our machine is Intel® Core™ i5-3340M CPU @ 2.70GHz, with 8GB RAM. Of the seven algorithms, DSMI, FIMUS, FEMI, (Grey + CPAANN), and DMI are comparable in computation time, although DSMI takes slightly less time than FIMUS, FEMI, (Grey + CPAANN) but more time than DMI. On the other hand, EMI and KNNI are significantly faster, especially for the big datasets. Although our algorithm needs higher computation time compared to EMI and KNNI, better imputation accuracy generally has a higher priority in missing value imputation [**20**]. We now analyse complexity for DSMI algorithm. We consider that there are *m*

attributes with missing values over the whole data set with n records, n_i records with one or more missing values, and n_c records ($n_c = n - n_i$) with no missing values. DSMI uses the C4.5 algorithm to build decision trees in the complete records n_c. The C4.5 algorithm has a complexity of $O(n_c m^2)$. Let, L is the maximum number of records in a leaf. The weighted similarity measure calculates in each leaf and the complexity is $O(n_c/L)$. Therefore, the overall complexity is $O(n_c/L(n_c m^2))$.

Table 2.19. Average execution time of different techniques (in seconds)

Dataset	DSMI	FIMUS	DMI	EMI	KNNI	FIMUS	FEMI	Grey+ CPAANN
Truck	1162.232	1181.41	1000.01	90.189	92.193	1181.41	1191.10	1378.61
Denver County	1001.301	1113.230	901.240	88.182	89.456	1113.230	1003.13	1110.10
Case	10.294	14.403	8.023	5.878	5.001	14.403	18.5	16.10
Individual	10.900	15.001	8,001	5.671	4.908	15.001	11.02	13.01

2.5 Conclusion

In this chapter, a brief overview of existing algorithms proposed by researchers for missing values imputation is presented. The literature review reveals that most research in this area is targeted at missing value imputation of numerical data and there is still a lack of research being done for missing values imputation of categorical data. We describe several recent missing value imputation algorithms that could handle categorical data and pointed out their deficiencies before introducing our proposed missing imputation method, called DSMI, to handle categorical missing values in traffic accident data. There are four stages in the DSMI algorithm. The algorithm first utilizes decision trees to find the set of correlated records. The missing values are then imputed from these records by exploiting the correlation between missing and non-missing

attributes within a record using the IS measure, as well as the direct and transitive correlations of attribute values across two records using a weighted similarity measure. Moreover, to handle the inherent uncertainty seen in real data, our algorithm imputes the missing values based on sampling from a list of potential imputed values based on their degree of affinity.

We presented extensive experimental results on four large publicly available traffic accident datasets, in which a large number of attributes are categorical. Comparisons with a number of state-of-the-art missing value imputation algorithms performed and our experiments indicated that the proposed algorithm significantly outperformed all existing algorithms.

Chapter 3: Noisy values Detection and Correction

This chapter describes the second part of the research, which is on noisy values detection and correction. The introduction about noisy values is discussed in Section 3.1. In Section 3.2, the existing well-known noisy values detection and correction algorithms are reviewed and analysed. Section 3.3 presents the proposed noisy values detection and correction algorithm NoiseCleaner in detail. Section 3.4 addresses the performances of NoiseCleaner and Section 3.5 draws the conclusions. The content of this chapter has largely been submitted for publication[1].

3.1 Introduction

Real world data are often corrupted by noise and are generally noisy and inconsistent [48]. This noise occurs due to errors in data collection, storage, and processing. If an organization does not take extreme care during data collection, then approximately 5% or more noisy data could be introduced to a dataset [69]. The detection and correction of noisy values are especially important in view of today's massive datasets, where the emphasis is often placed on the volume rather than the quality of the data. Noisy data badly affect the results of any data analysis. The presence of noisy values reduces the quality of the analysis models learned from the data and weakens their predictive or descriptive performance. Moreover, these analysis models would become overly complex in order to accommodate such noise.

[1] Rupam Deb, Alan Wee-Chung Liew, "Noisy Values Detection and Correction of Traffic Accident Data", submitted to Information Sciences.

In order to improve the quality of the traffic accident data, raw data is often preprocessed and cleansed before being analysed. Identifying and correcting noisy values is an important goal of data cleansing and preprocessing.

In this chapter, we proposed a novel data cleansing method called NoiseCleaner for detecting noisy attributes values and predicting their correct values on traffic accident datasets.

3.2 Literature review

Identification of noisy data is an important data preprocessing task for improving data quality. Many noise detection algorithms have been proposed for various applications [3, 16, 58, 70-74, 86-89]. Among them, HCleaner [74], NOISERANK [72], Polishing method [73] and Error Detection and Impact-sensitive instance Ranking (EDIR) [16] are some well-known noisy value detection algorithms.

In HCleaner [74], data objects that are irrelevant or only weakly relevant are considered as noise. HCleaner is based on the concept of hyperclique patterns which consist of objects that are strongly similar to each other. In HCleaner, every pair of objects within a hyperclique pattern is guaranteed to have a cosine similarity above a certain threshold. 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 [73] 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 assumes that there is some pattern of relationship among the different components of a dataset. Therefore, except for totally irrelevant attributes, each attribute would at least be related to some extent to the target class. 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: the prediction stage and the 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.* [72] proposed 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 could 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 the 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 into 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 noisiest 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 the 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) [16] locates erroneous instances and attributes and ranks suspicious instances based on their impact on system performance. At first, EDIR trains a benchmark classifier *T* from the noisy dataset *D*. The instances that cannot be classified by *T* are treated as suspicious and forwarded to a subset *S*. Each instance contains *n* attributes A_1, A_2, …, A_n and each attribute A_i has V_i possible values. To rank instances in *S*, EDIR uses an impact measure based on the information-gain ratio. Like the polishing method, EDIR also changes the attribute value to correctly classify the record. However, EDIR differs from the polishing method in several aspects. Unlike the 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 combinations of the two or three values.

The co-appearance based Analysis for Incorrect Records and Attribute-values Detection (CAIRAD) [71] 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 [70] uses the kNN technique to classify a record. At first, the dataset is divided into training and testing datasets. After that, RDCL classifies the record in the testing dataset by using the majority class of its kNN records in the training dataset. RDCL identifies suspicious records, whereas the polishing technique and EDIR detect both noisy attributes values and records.

In many practical scenarios, it is assumed 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.

3.3 Proposed NoiseCleaner algorithm

We proposed a data cleansing algorithm called NoiseCleaner for the traffic accident data. In this approach, at first attribute values with occurrence frequency of 1 or 2 are flagged as potentially suspicious (It is very uncommon that typographical errors repeated more than 2 times for a value and that's why value's frequency less than 3 is flagged as suspicious.). We then correct for simple typographical errors that occur in the flagged values using Levenshtein distance [48]. 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 NoiseCleaner, Table 3.1 shows a sample traffic accident dataset (bold text values of Table 3.1 are marked as suspicious noisy values). Subsets are built based on the suspicious noisy values ('Active' and 'Fog'). These subsets are shown in Tables 3.2-3.5.

Table 3.1. 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 3.2. 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 3.3. 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 3.4. 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 3.5. Subset S_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 weighted similarity 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 weighted similarity 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'.

3.3.1 P-measure

To calculate the *P-measure*, the two subsets S_s and S_d are created based on the suspicious noisy attribute value x of record *r* is considered. 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

$$P_s + P_{sp} + P_{dp} = 1 \qquad (11)$$

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

$$P_s = k_1 \times P_{sp} = k_1 \times k_2 \times P_{dp} \quad (12)$$

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

$$P_s = \frac{k_1 \times k_2}{k_1 \times k_2 + k_2 + 1} \quad (13)$$

$$P_{sp} = \frac{k_2}{k_1 \times k_2 + k_2 + 1} \quad (14)$$

$$P_{dp} = \frac{1}{k_1 \times k_2 + k_2 + 1} \quad (15)$$

Substituting k_1 and k_2 in equations (14) and (15) we get,

$$P_{sp} = \frac{(N_{S_s} - N_x)(K-1)}{N_{S_s}(K-1) + \sum_{i=1}^{N_{S_d}} \delta_i} \quad (16)$$

$$P_{dp} = \frac{\sum_{i=1}^{N_{S_d}} \delta_i}{N_{S_s}(K-1) + \sum_{i=1}^{N_{S_d}} \delta_i} \quad (17)$$

Next, we 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

$$P_{sp}(y_k) = \frac{N_{y_k}(K-1)}{N_{S_s}(K-1)+\sum_{i=1}^{N_{S_d}} \delta_i} \qquad (18)$$

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

$$P_{dp}(z_k) = \frac{\sum_{j=1}^{N_{z_k}} \gamma_j}{N_{S_s}(K-1)+\sum_{i=1}^{N_{S_d}} \delta_i} \qquad (19)$$

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 (12), (18), and (19), based on Tables 3.4 and 3.5. 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 3.2 and 3.3.

Once *P-measure* is computed and the suspicious noisy attribute value cannot be declared as correct, the weighted similarity measure of the suspicious noisy attribute value with the other possible values of the attribute in both S_s and S_d are computed. The weighted similarity 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 weighted similarity measure between 'Fog' and 'Bad', 'Fog' and 'Stormy', 'Fog' and 'Rainy'. The next section explains how the weighted similarity measure between two attribute values is computed.

3.3.2 Weighted similarity measure

Weighted similarity measure, S_{ij} (this measure is described in details at Section 2.3.2) 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). 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

$$S_{ij} = C_1 \times S'_{ij} + C_2 \times S''_{ij} \qquad (20)$$

To illustrate how weighted similarity measure is computed for noisy values detection and correction, we will use the records in Table 3.6 with vertex nodes labelled from 1 to 8 as an example. The 1st and 2nd levels weighted similarity measure graphs are constructed with respect to 'weather condition' attribute in Figure 3.1 and Figure 3.2.

Table 3.6. Aggregated table (Tables 3.4 and 3.5) 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)

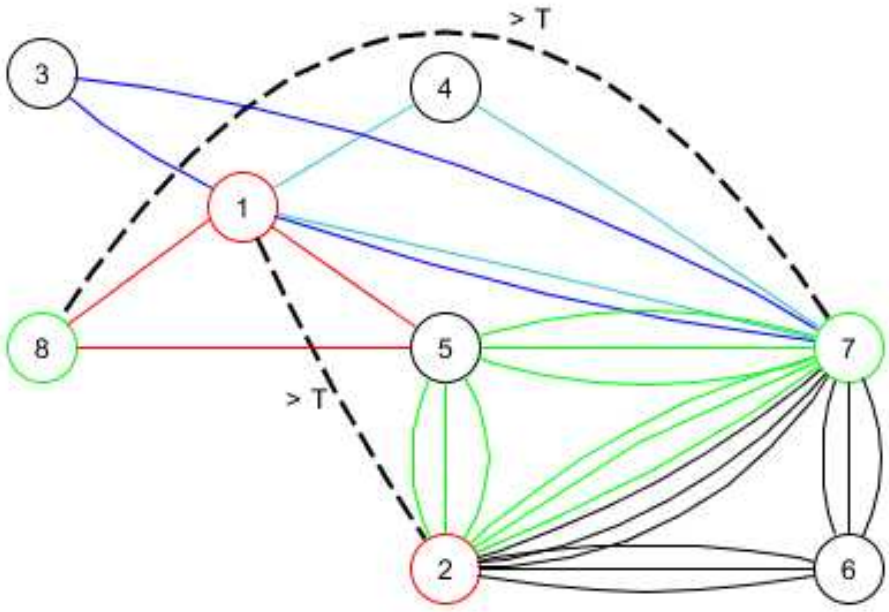


Figure 3.1: 1st level weighted similarity measure graph with respect to 'weather condition' attribute constructed from Table 3.6

From the graph in Figure 3.1, it shows 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 (Section 3.4.3 gives explanation for this choice). In Figure 3.1, we connect nodes having 1st level similarities greater than the threshold T by dotted lines [17, 56]. For easier visualization of the vertices within the same record and their associated edges, different colours are used.

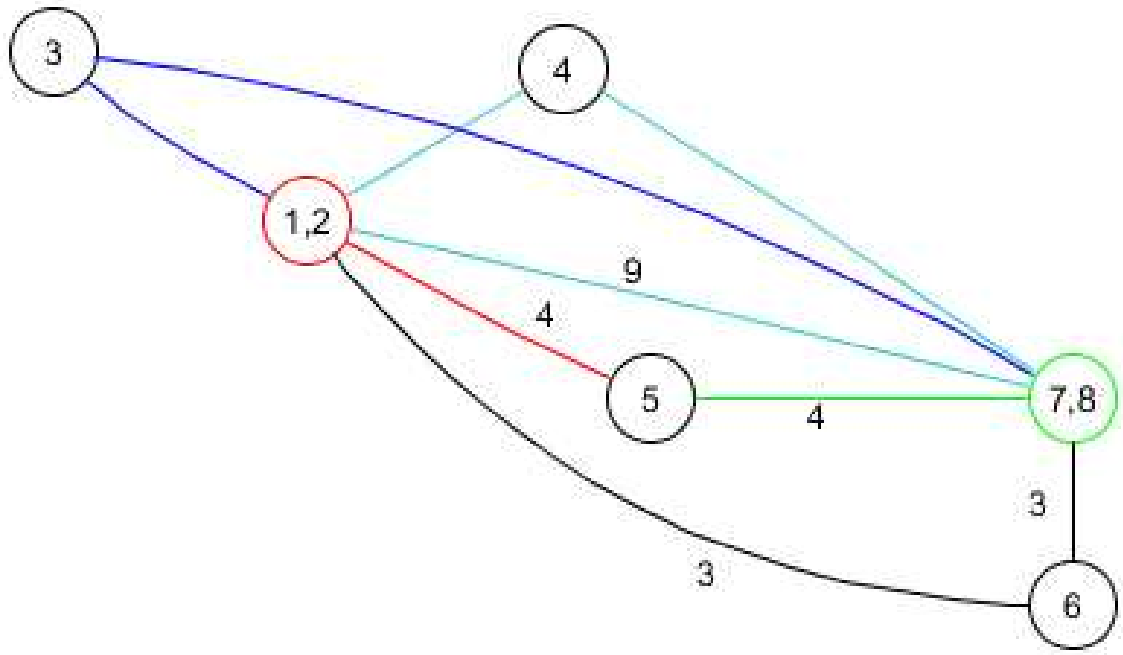


Figure 3.2: 2nd level weighted similarity measure graph with respect to 'weather condition' attribute constructed from Figure 3.1

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 3.2. Here, vertices 1 and 2 are merged because $S'_{1,2} > T$ and vertices 7 and 8 are also merged. In Figure 3.2, 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 (6) 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 weighted similarity 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

$$x = \begin{cases} y & \text{if } (P_{sp}(y) \text{ or } P_{dp}(y)) > P_s(x) \\ x & \text{otherwise} \end{cases} \qquad (21)$$

By considering both the weighted similarity 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'.

3.3.3 NoiseCleaner algorithm

The proposed algorithm is summarized below.

NoiseCleaner Algorithm:

1. Count 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 (13)

3.2.2 FOR each distinct value y_k of *X* from S_s, calculate $P_{sp}(y_k)$ using Equation (18)

3.2.3 FOR each distinct value z_k of *X* from S_d, calculate $P_{dp}(z_k)$ using Equation (19)

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 (20)

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 (21)

3.4 Experimental results and discussion

3.4.1 Datasets

We performed experiment on four datasets. Two road crash datasets are taken from the State of Queensland, Australia [75], and other two datasets are taken from the motor vehicle crashes of the New York State, United States [76]. In the four datasets, most of the attributes are categorical. We listed the four datasets in Table 3.7.

Table 3.7. Description of datasets

Abbreviation	Dataset name	#Records	#Categorical attributes	#Numerical attributes	As on date
RCL	Road Crash Locations (RCL) [75]	251705	30	20	31 July, 2015
RC	Road Casualties (RC) [75]	15744	6	1	31 July, 2015
MCI	Motor vehicle crash-Case Information: 2011 (MCI) [76]	13889	17	1	24 September, 2014
MII	Motor vehicle crash-Individual Information: 2011 (MII) [76]	17858	11	3	24 September, 2014

3.4.2 Performance measures

Several performance measures are commonly used to evaluate the performance of noisy value detection algorithms [71, 72]. A popular measure is *precision*. *Precision* is defined as the ratio of the number of relevant records retrieved to the total number of irrelevant and relevant records retrieved as in Equation (23)

$$\boldsymbol{Precision} = \frac{\boldsymbol{number\ of\ true\ noisy\ instances\ detected}}{\boldsymbol{number\ of\ all\ instances\ identified\ as\ noisy}} \qquad (23)$$

Another useful measure is *recall*. *Recall* is defined as the ratio of the number of relevant records retrieved to the total number of relevant records in the database as in Equation (24)

$$\mathbf{Recall} = \frac{\mathbf{number\ of\ true\ noisy\ instances\ detected}}{\mathbf{number\ of\ all\ noisy\ instances\ in\ the\ dataset}} \qquad (24)$$

Precision and *Recall* are inversely related. *F-measure* is defined as the harmonic mean of *precision* and *recall* as in Equation (25).

$$\boldsymbol{F} = \mathbf{2}\,\frac{(\boldsymbol{precision} \times \boldsymbol{recall})}{(\boldsymbol{precision} + \boldsymbol{recall})} \qquad (25)$$

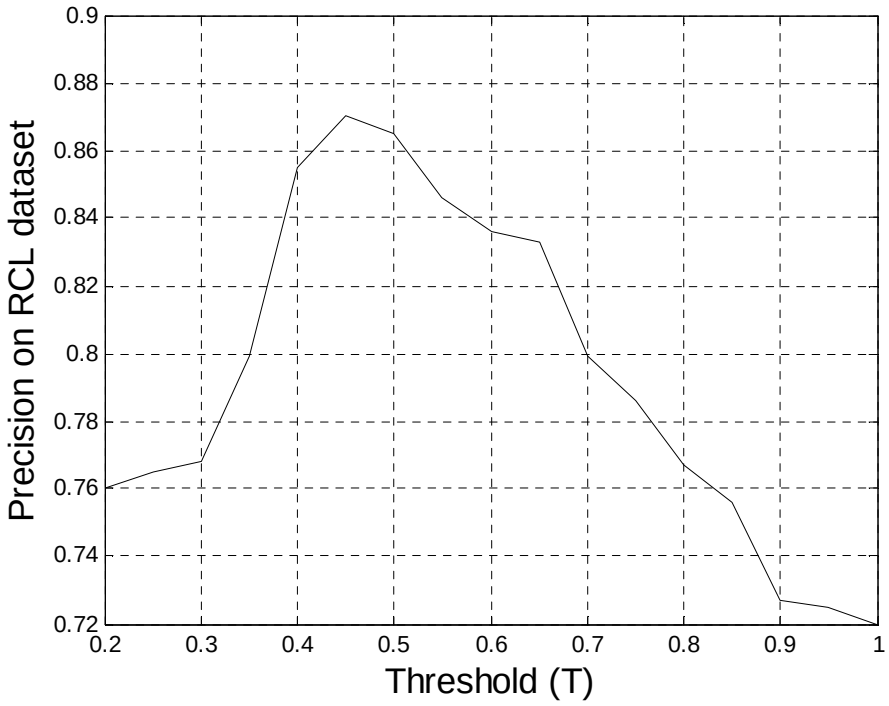
Equation (25) is also known as the $\boldsymbol{F_1}$ measure or traditional F-measure or balanced F-score, because *recall* and *precision* are weighted evenly. It is possible to give emphasis to precision or recall by using

$$\boldsymbol{F_\beta} = (\mathbf{1} + \boldsymbol{\beta^2})\,\frac{(\boldsymbol{precision} \times \boldsymbol{recall})}{\left((\boldsymbol{\beta^2} \times \boldsymbol{precision}) + \boldsymbol{recall}\right)} \qquad (26)$$

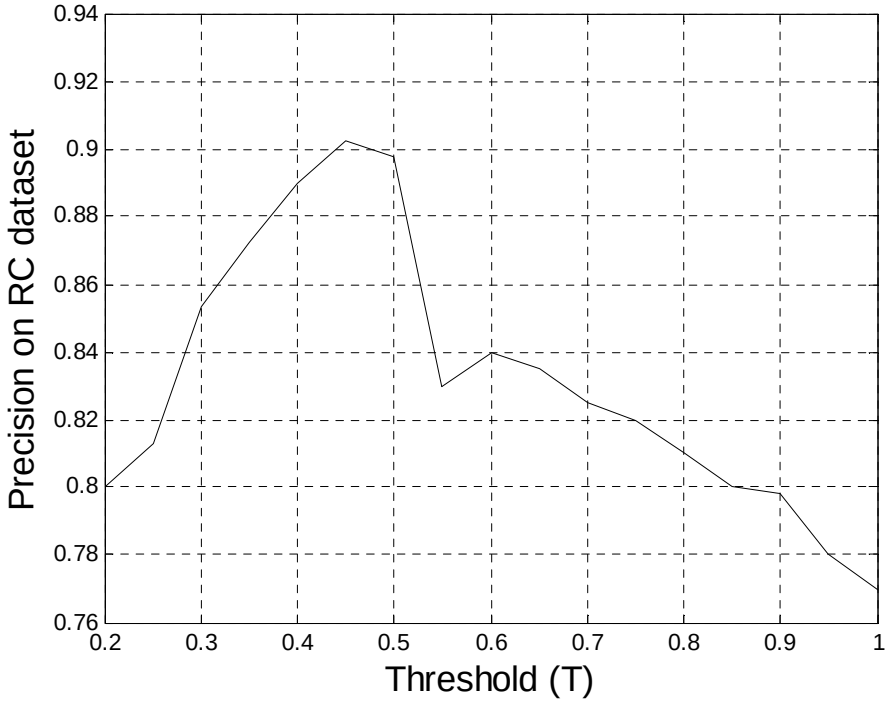
By setting the parameter $\boldsymbol{\beta}$ ($\boldsymbol{\beta} > \mathbf{0}$) the user can assign more importance to either *precision* or *recall* in the computation of the *F-measure*. For example, $\boldsymbol{F_2}$ weighs *recall* higher than *precision*, and $\boldsymbol{F_{0.5}}$ puts more emphasis on *precision* than *recall*. To detect the noisy values and evaluate the noise detection performance, it is important to know how many noisy values are identified out of the total detected noisy values by the algorithm.

3.4.3 Parameter selection for weighted similarity measure

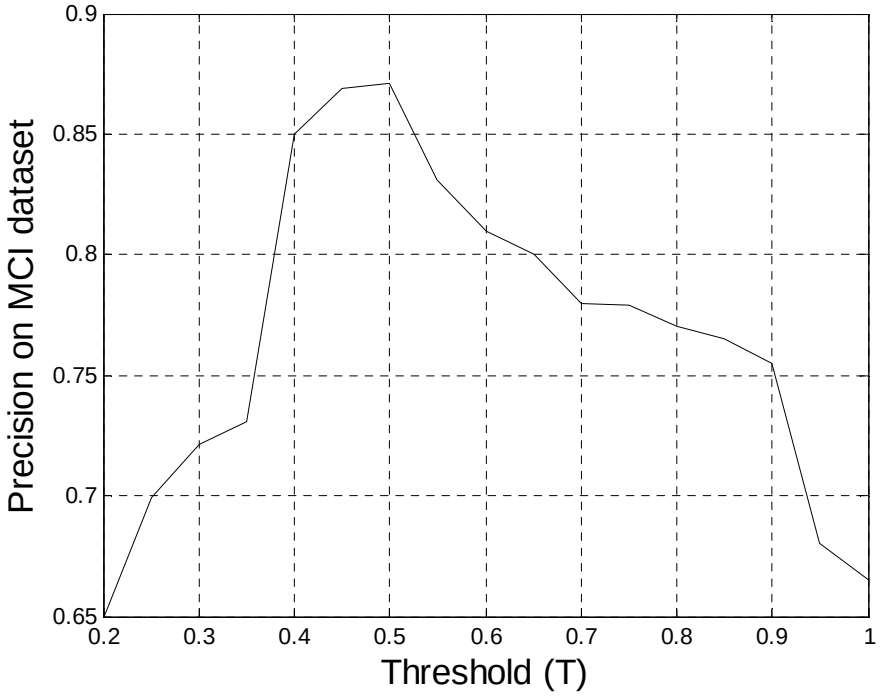
Weighted similarity measure requires the setting of three parameters: T, C_1, and C_2. Using the precision measure, we analyse the four datasets to select the best threshold parameter T. The result is shown in Figure 3.3 and we set T=0.45. In Table 3.8, we see that the optimum values for C_1, and C_2 are 0.65 and 0.35, respectively.



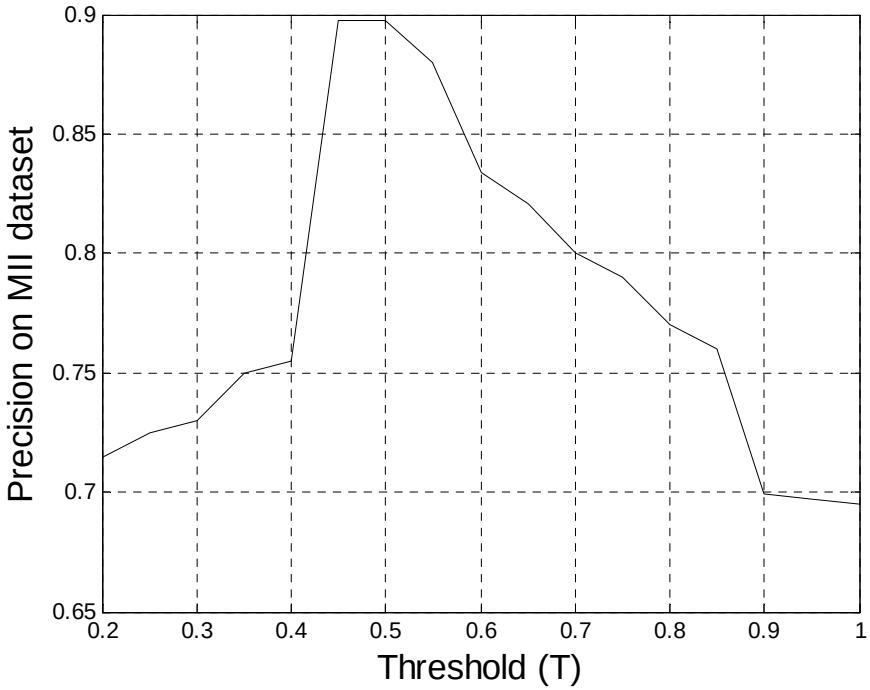
(a) Threshold parameter T on RCL dataset



(b) Threshold parameter T on RC dataset



(c) Threshold parameter T on MCI dataset



(d) Threshold parameter T on MII dataset

Figure 3.3: Threshold parameter T on four datasets

Table 3.8. C_1 and C_2 parameters 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

3.4.4 Noisy values simulation

We use four types of noisy patterns in our test datasets: simple, medium, complex, and blended [3, 17]. 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 these experiments, 32 noisy combinations (4 noisy ratios × 4 noisy patterns × 2 noisy models) are created. For each combination, five datasets i.e. in total we create 160 noisy datasets (32 combinations × 5 datasets per combination) are generated for each real dataset as shown in Table 3.9.

Table 3.9. Noisy value simulation

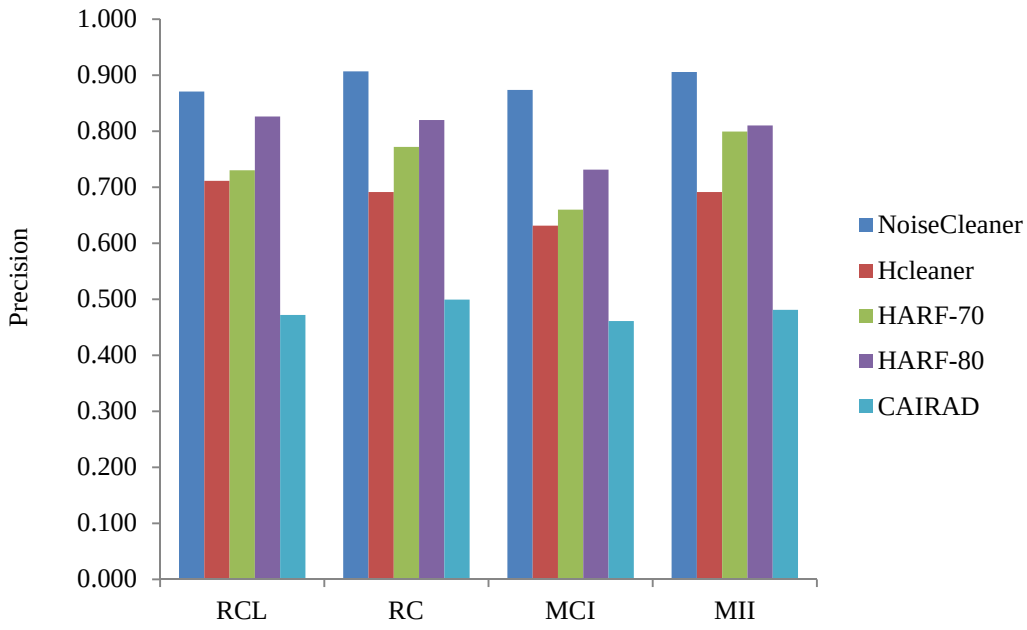
Noisy patterns	Number of attributes having noisy values in a record	Typographical error	Noise ratios	Noisy model	Number of datasets for each pattern
Simple Medium Complex Blended	1 Up to 50% Up to 80% Simple-25%, Medium-50% and Complex-25%	1% with randomly 1, 2, or 3 characters	2%, 4%, 6% and 8%	Overall and Uniformly distributed	5

3.4.5 Experimental results

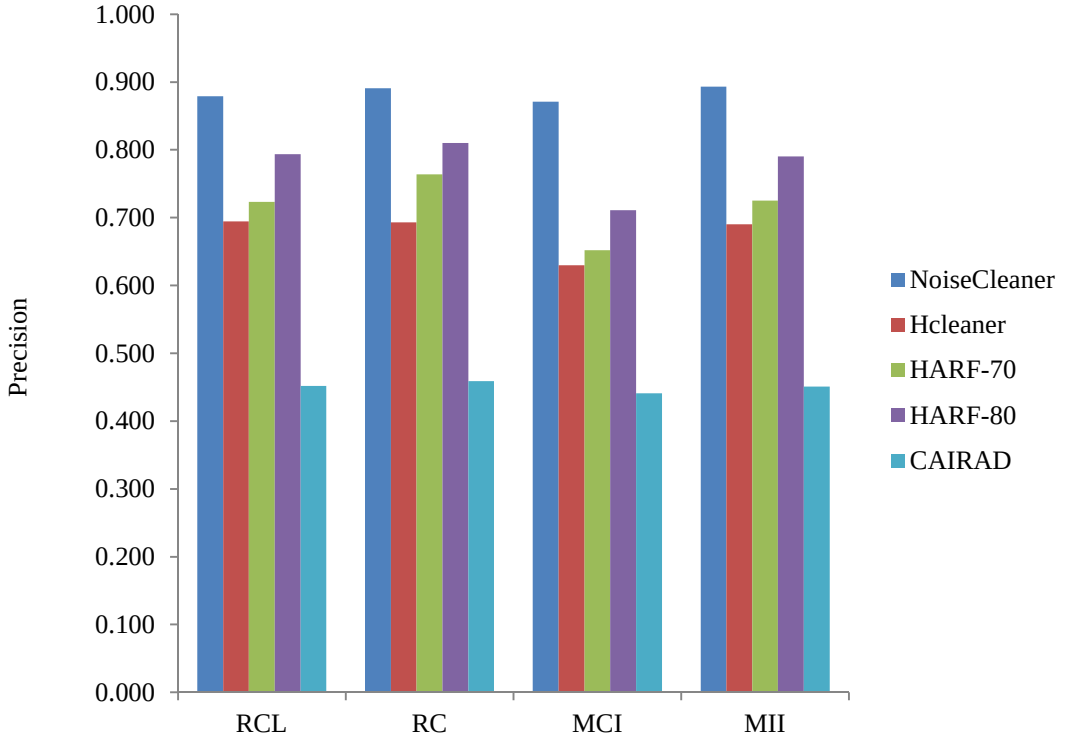
The proposed algorithm NoiseCleaner compares with three noisy values identification methods NOISERANK [72], HCleaner [74], and CAIRAD [71]. In NOISERANK

algorithm, there are several ensemble methods. Among them, HARF-80 and HARF-70 variants performed best compared to the other variants. These two variants are taking into consideration to compare with the NoiseCleaner algorithm. Simple typographical errors are corrected using Levenshtein distance as a pre-processing for all methods.

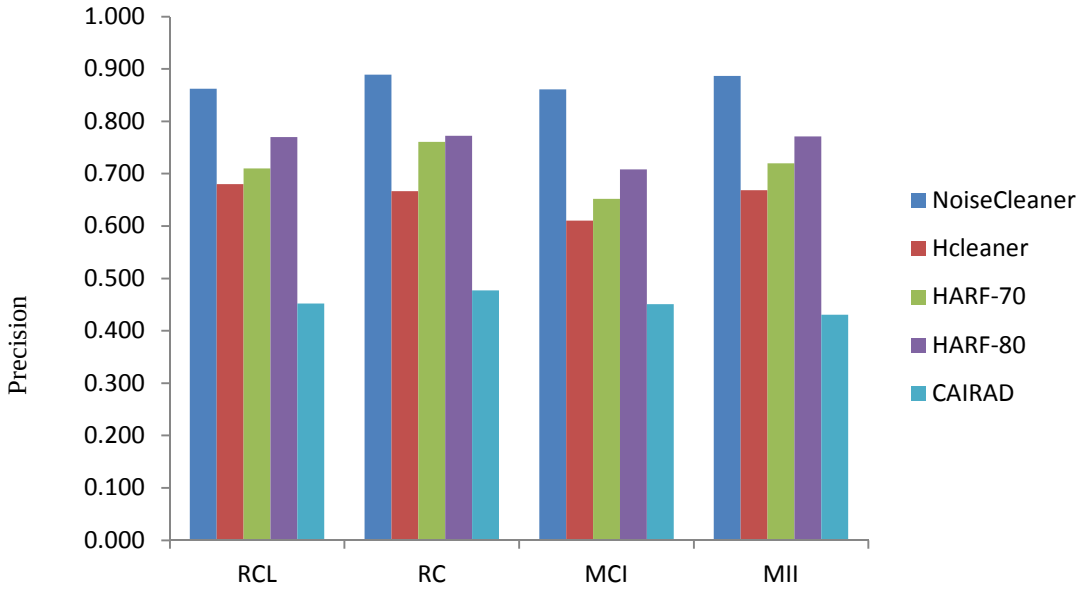
The noisy values detection accuracy (*Precision*) of NoiseCleaner, HCleaner, HARF-70, HARF-80, and CAIRAD is presented in Figure 3.4. For each dataset, four combinations of four noisy ratios (2%, 4%, 6%, and 8%), and one noisy pattern (medium) are shown. Figure 3.5 shows the corresponding *recall* result and Figure 3.6 shows the $F_{0.5}$ result. From these results, it can be seen that NoiseCleaner performs significantly better than HARF-80, HARF-70, HCleaner, and CAIRAD on all 4 datasets.



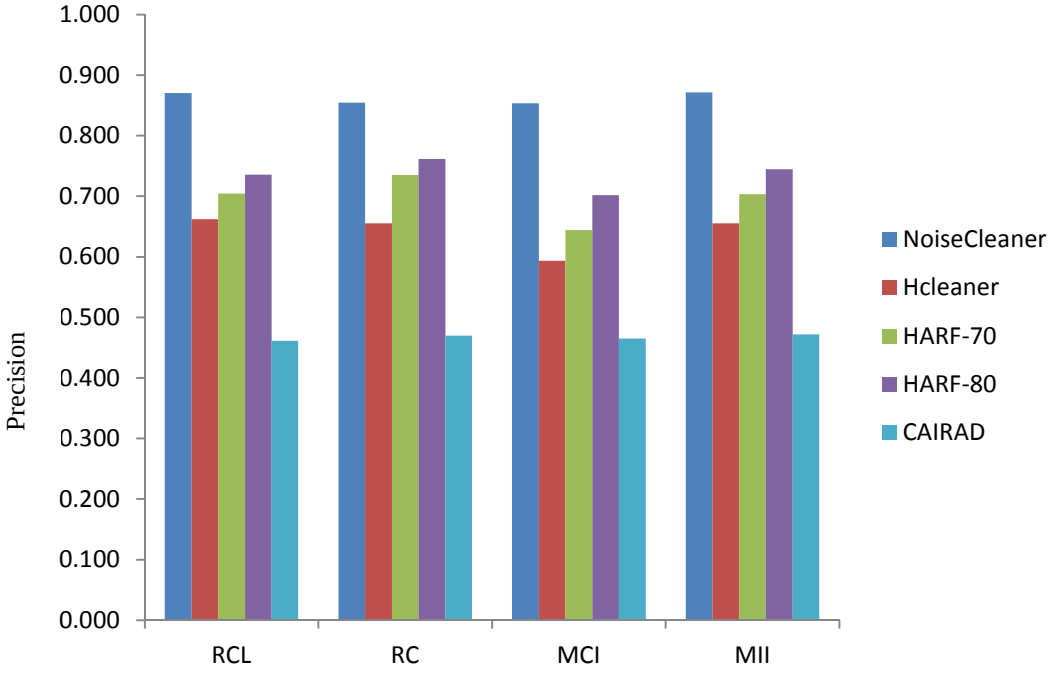
(a) 2% injected noise



(b) 4% injected noise

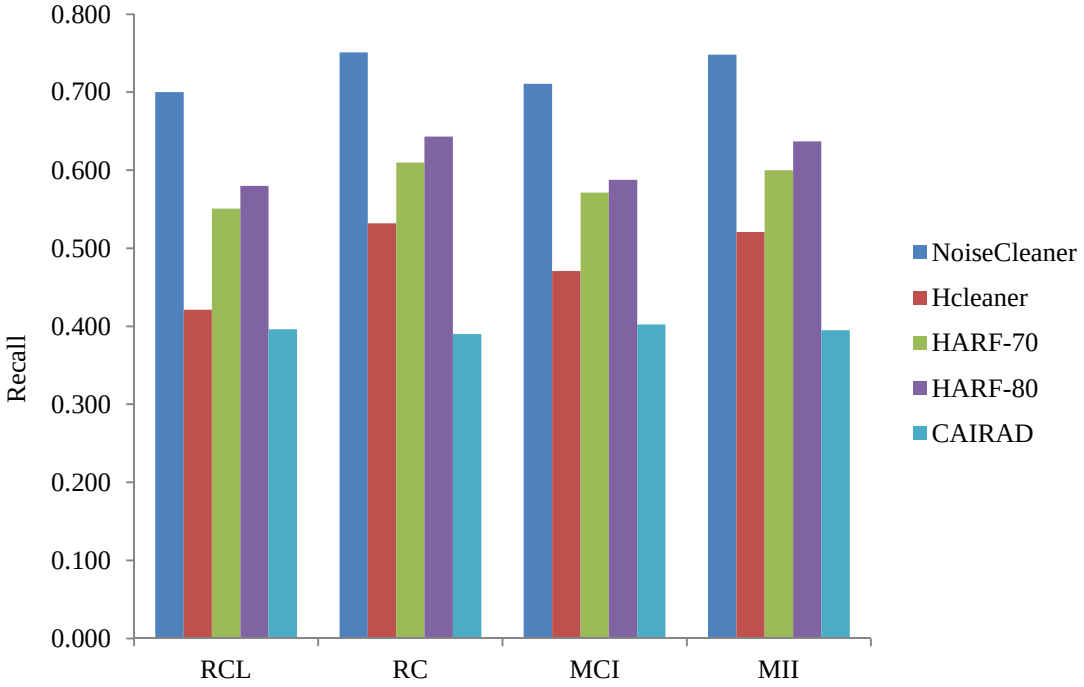


(c) 6% injected noise

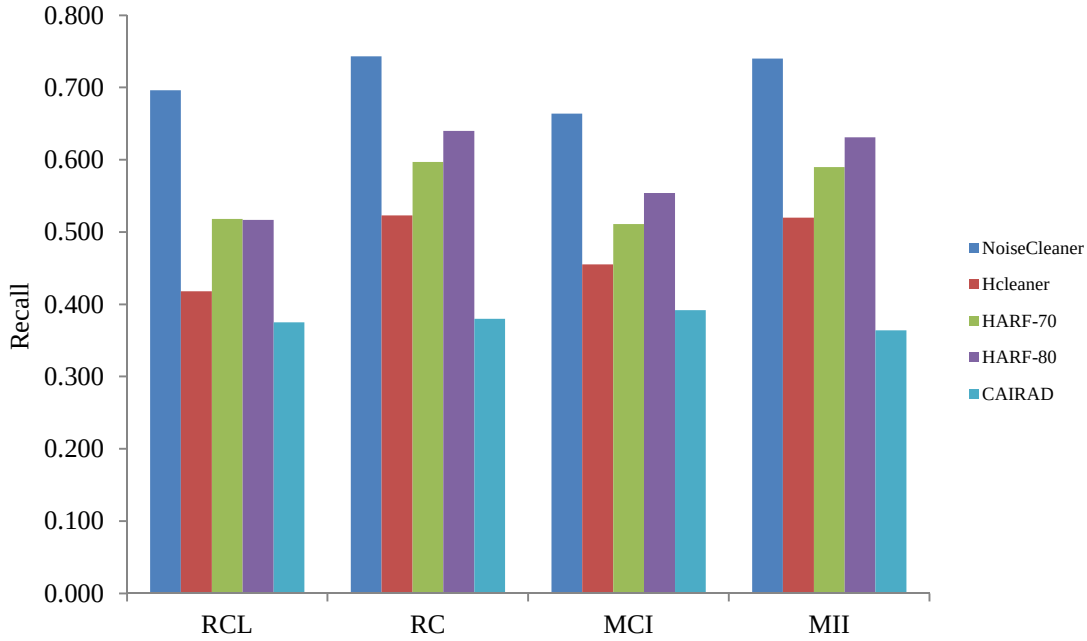


(d) 8% injected noise

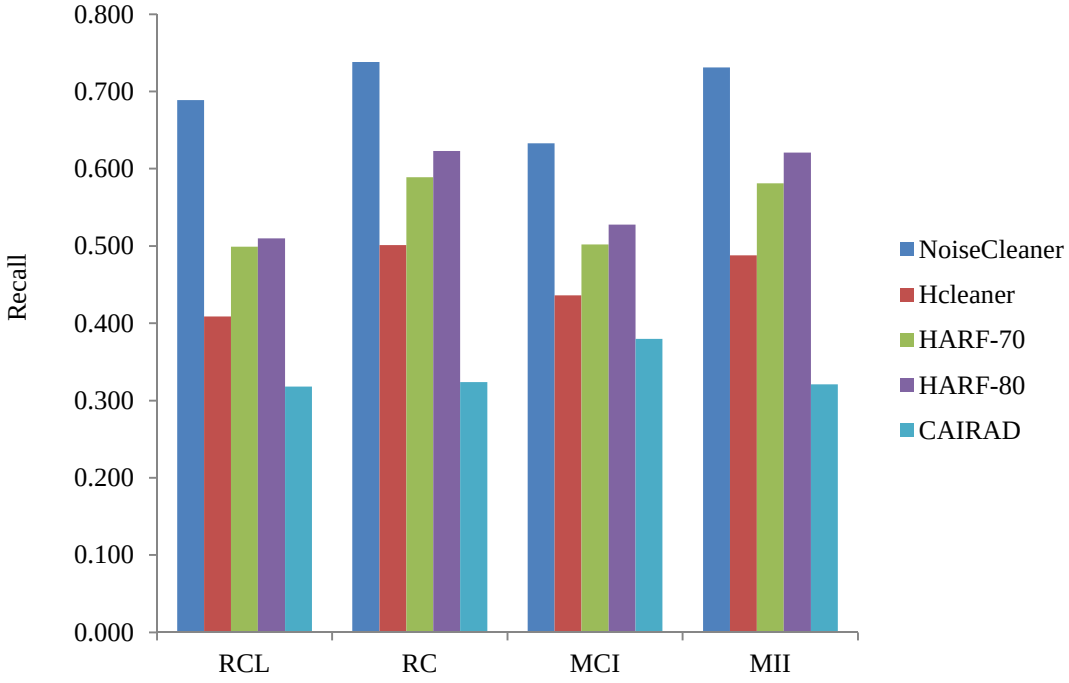
Figure 3.4. The *precision* results of different noisy value detection algorithms on 4 datasets for various 'noisy ratios' and 'medium noisy pattern'.



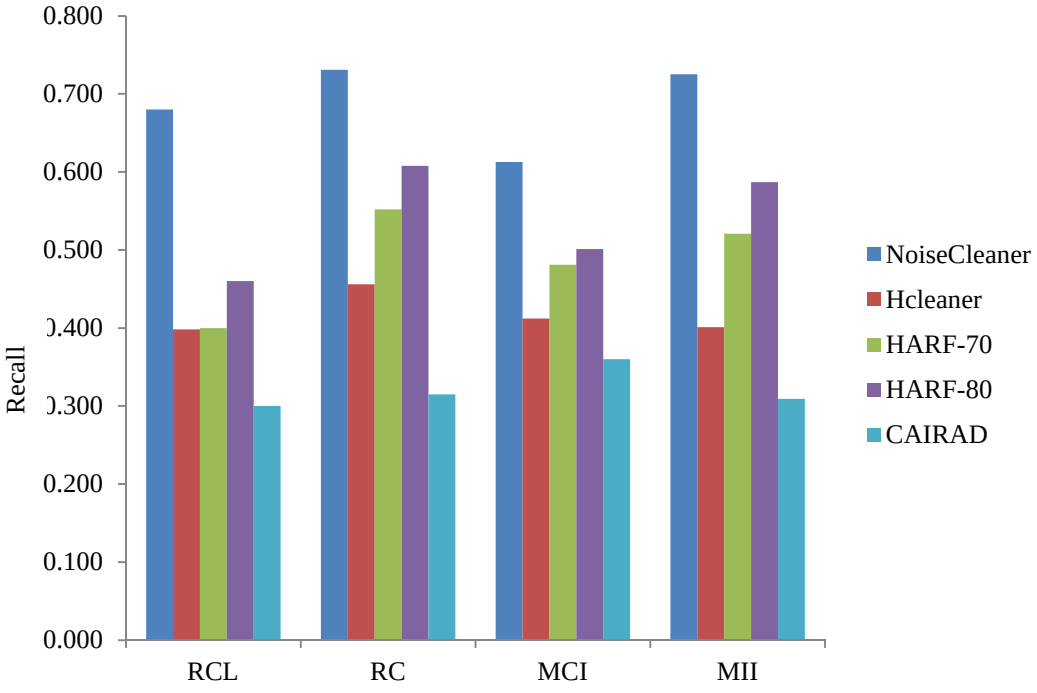
(a) 2% injected noise



(b) 4% injected noise

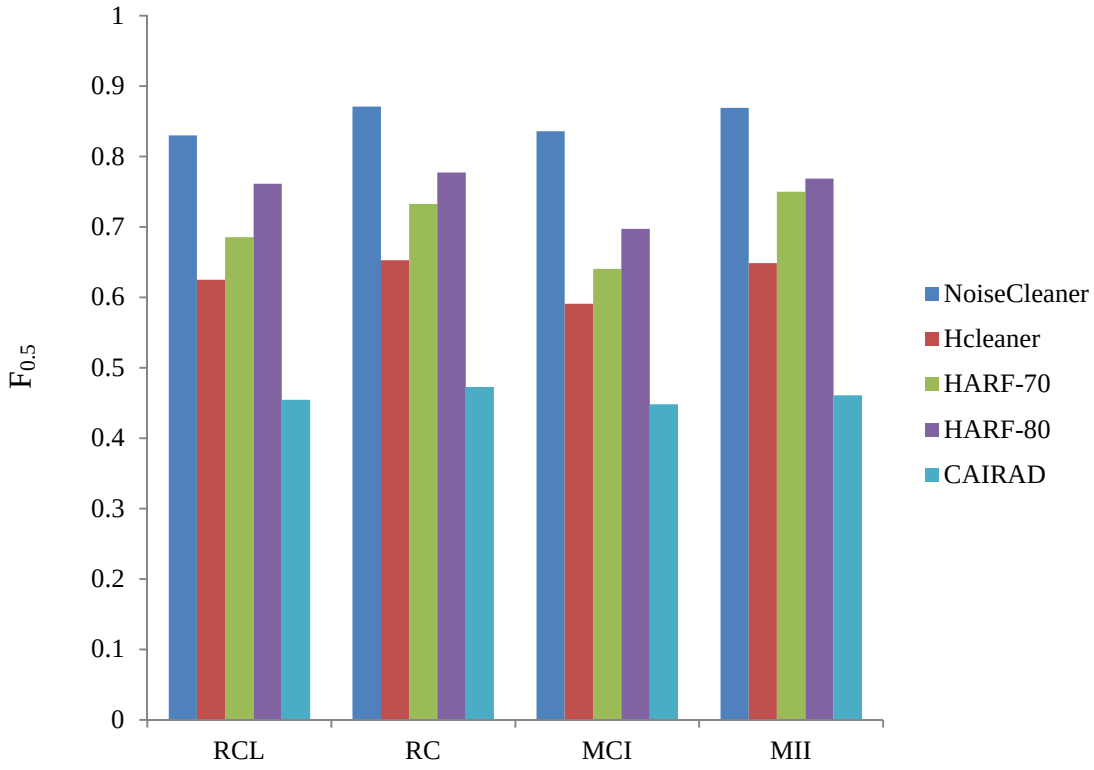


(c) 6% injected noise

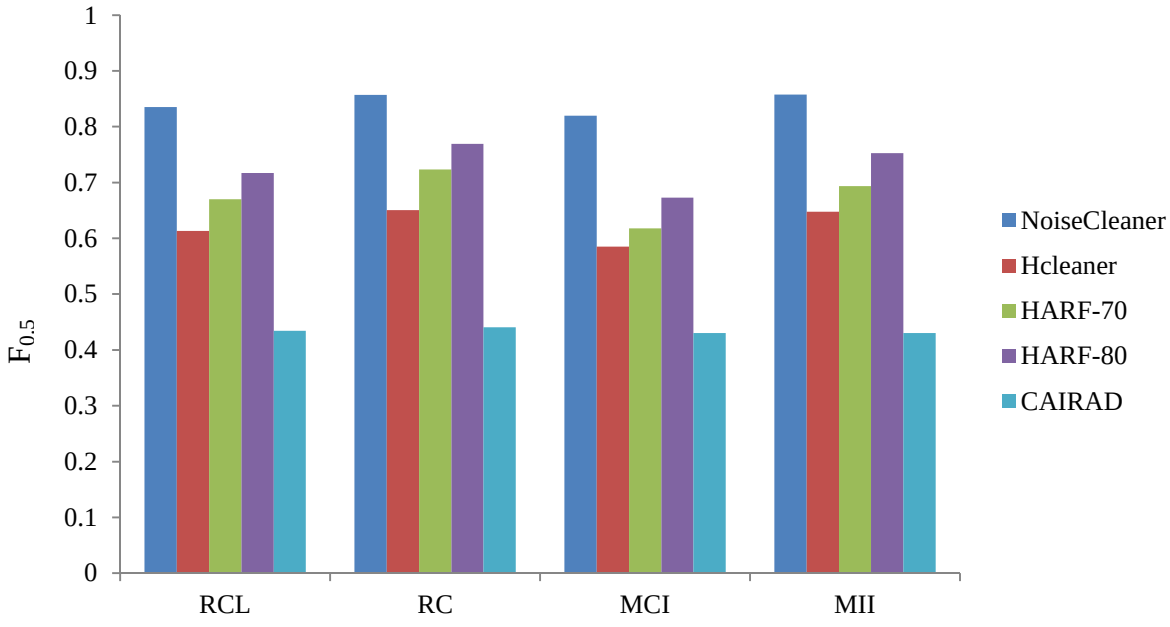


(d) 8% injected noise

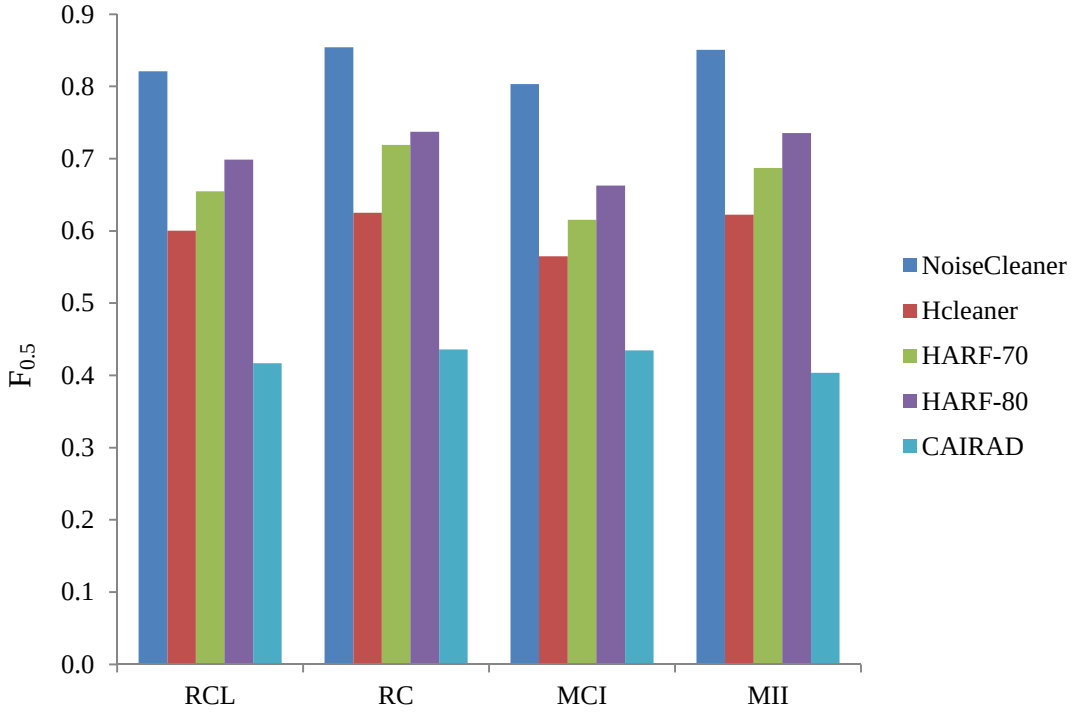
Figure 3.5. The *recall* results of different noisy value detection algorithms on 4 datasets for various ‘noisy ratios’ and ‘medium noisy pattern’.



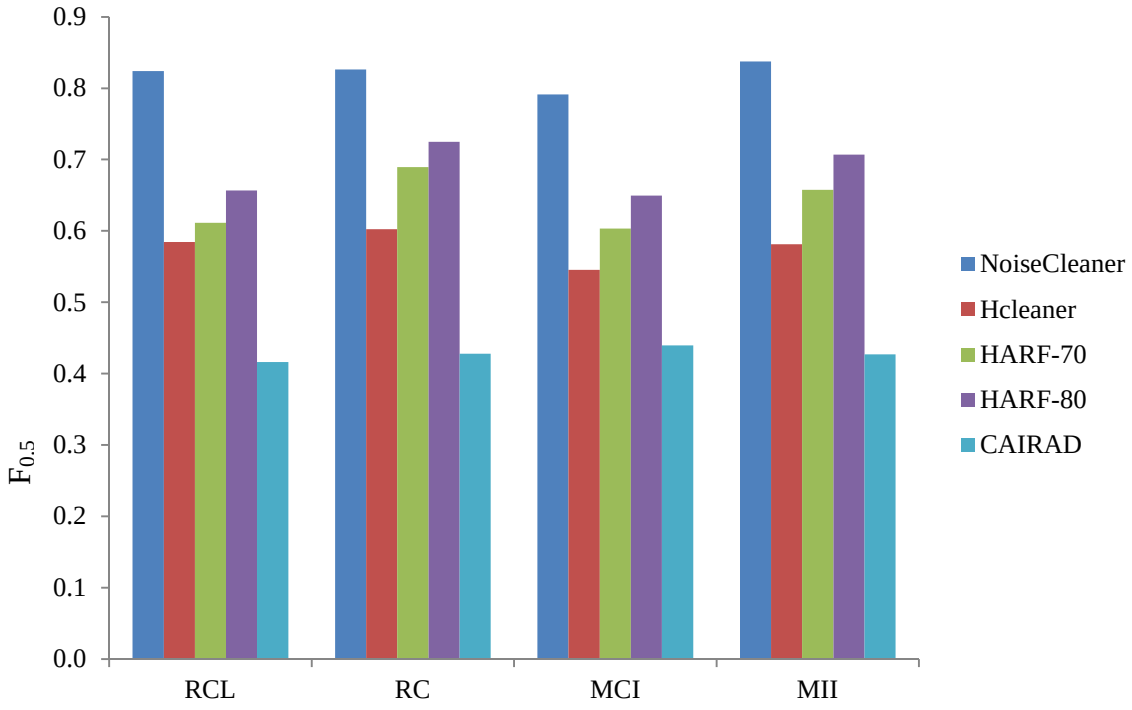
(a) 2% injected noise



(b) 4% injected noise



(c) 6% injected noise



(d) 8% injected noise

Figure 3.6. The $F_{0.5}$ results of different noisy value detection algorithms on 4 datasets for various 'noisy ratios' and 'medium noisy pattern'.

The noisy value detection accuracy of NoiseCleaner, HARF-70, HARF-80, and HCleaner on the 4 datasets in terms of *precision* for the 32 noisy combinations is presented in Tables 3.10-3.13. 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, 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 3.11).

Table 3.10. Performance on RC dataset using Precision

Noise combination			Precision				
			NoiseCleaner	HARF-80	HARF-70	HCleaner	CAIRAD
2%	Overall	Simple	**0.9087±0.01**	*0.8233±0.02*	0.7732±0.02	0.6896±0.01	0.4980±0.02
		Medium	**0.9084±0.02**	*0.8246±0.01*	0.7752±0.03	0.6891±0.02	0.4980±0.02
		Complex	**0.9082±0.01**	*0.8144±0.04*	0.7688±0.01	0.6895±0.01	0.4957±0.02
		Blended	**0.9078±0.03**	*0.8135±0.05*	0.7678±0.02	0.6891±0.04	0.4951±0.03
	UD	Simple	**0.9065±0.01**	*0.8237±0.03*	0.7776±0.02	0.6898±0.01	0.4901±0.02
		Medium	**0.9062±0.04**	*0.8245±0.01*	0.7728±0.04	0.6865±0.04	0.4900±0.02
		Complex	**0.9059±0.02**	*0.8189±0.05*	0.7701±0.03	0.6894±0.05	0.4886±0.03
		Blended	**0.9047±0.01**	*0.8178±0.02*	0.7709±0.05	0.6898±0.01	0.4880±0.04
4%	Overall	Simple	**0.8901±0.01**	*0.8128±0.02*	0.7662±0.01	0.6918±0.01	0.4845±0.02
		Medium	**0.8910±0.03**	*0.8119±0.03*	0.7669±0.01	0.6904±0.03	0.4843±0.01
		Complex	**0.8912±0.02**	*0.8127±0.04*	0.7661±0.03	0.6808±0.04	0.4842±0.03
		Blended	**0.8903±0.01**	*0.8110±0.03*	0.7664±0.01	0.6802±0.02	0.4840±0.03
	UD	Simple	**0.9001±0.02**	*0.8104±0.01*	0.7663±0.02	0.6887±0.01	0.4745±0.03

		Medium	**0.8900±0.04**	*0.8061±0.02*	0.7648±0.01	0.6745±0.06	0.4739±0.02
		Complex	**0.8923±0.03**	*0.8098±0.05*	0.7646±0.05	0.6732±0.02	0.4730±0.01
		Blended	**0.8921±0.01**	*0.8040±0.06*	0.7642±0.01	0.6731±0.03	0.4704±0.02
6%	Overall	Simple	**0.8910±0.01**	*0.8032±0.02*	0.7640±0.03	0.6702±0.02	0.4730±0.02
		Medium	**0.8909±0.02**	*0.8031±0.02*	0.7631±0.01	0.6688±0.04	0.4729±0.01
		Complex	**0.8806±0.06**	*0.7692±0.05*	0.7638±0.04	0.6650±0.02	0.4720±0.02
		Blended	**0.8801±0.02**	*0.7693±0.04*	0.7620±0.01	0.6615±0.04	0.4720±0.03
	UD	Simple	**0.8802±0.01**	*0.7793±0.03*	0.7620±0.04	0.6608±0.02	0.4718±0.04
		Medium	**0.8801±0.07**	*0.7692±0.02*	0.7578±0.02	0.6602±0.06	0.4710±0.01
		Complex	**0.8800±0.04**	*0.7691±0.01*	0.7540±0.08	0.6604±0.05	0.4709±0.02
		Blended	**0.8800±0.02**	*0.7580±0.06*	0.7400±0.10	0.6568±0.01	0.4708±0.03
8%	Overall	Simple	**0.8901±0.02**	*0.7621±0.03*	0.7400±0.01	0.6672±0.03	0.4710±0.02
		Medium	**0.8886±0.04**	*0.7605±0.04*	0.7379±0.03	0.6648±0.02	0.4708±0.03
		Complex	**0.8885±0.01**	*0.7590±0.06*	0.7358±0.05	0.6535±0.06	0.4707±0.04
		Blended	**0.8880±0.05**	*0.7538±0.07*	0.7356±0.06	0.6522±0.07	0.4706±0.01
	UD	Simple	**0.8880±0.03**	*0.7910±0.02*	0.7301±0.07	0.6610±0.07	0.4705±0.02
		Medium	**0.8879±0.05**	*0.7903±0.05*	0.7280±0.04	0.6504±0.03	0.4700±0.03
		Complex	**0.8877±0.03**	*0.7940±0.03*	0.7020±0.04	0.6501±0.04	0.4700±0.04
		Blended	**0.8872±0.06**	*0.7945±0.07*	0.7210±0.05	0.6500±0.02	0.4880±0.02

Table 3.11. Performance on MII dataset using Precision

			Precision				
Noise combination			NoiseCleaner	HARF-80	HARF-70	HCleaner	CAIRAD
2%	Overall	Simple	**0.9060±0.01**	0.8108±0.02	*0.8120±0.01*	0.6935±0.02	0.4896±0.02
		Medium	**0.9058±0.03**	0.8103±0.01	*0.8030±0.03*	0.7030±0.04	0.4894±0.01

		Complex	**0.9057±0.04**	*0.8111±0.03*	0.8015±0.03	0.7036±0.05	0.4892±0.04
		Blended	**0.9055±0.06**	*0.8110±0.02*	0.8004±0.02	0.7040±0.02	0.4890±0.03
	UD	Simple	**0.9050±0.02**	0.8014±0.04	*0.8019±0.04*	0.7114±0.05	0.4889±0.04
		Medium	**0.9048±0.01**	0.8024±0.05	*0.8091±0.05*	0.7018±0.10	0.4888±0.08
		Complex	**0.9047±0.03**	*0.7929±0.06*	0.7845±0.06	0.6922±0.05	0.4888±0.03
		Blended	**0.9044±0.04**	*0.7930±0.07*	0.7231±0.03	0.6910±0.04	0.4887±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
		Medium	**0.8943±0.03**	*0.7981±0.03*	0.7302±0.03	0.6887±0.02	0.4891±0.02
		Complex	**0.8942±0.02**	*0.7850±0.05*	0.7240±0.04	0.6780±0.04	0.4888±0.03
		Blended	**0.8940±0.04**	*0.7895±0.01*	0.7148±0.05	0.6781±0.06	0.4883±0.02
	UD	Simple	**0.8935±0.05**	*0.7896±0.06*	0.7223±0.04	0.6878±0.02	0.4884±0.01
		Medium	**0.8934±0.06**	*0.7888±0.04*	0.7215±0.03	0.6858±0.07	0.4878±0.02
		Complex	**0.8933±0.07**	*0.7864±0.01*	0.7122±0.02	0.6796±0.09	0.4870±0.04
		Blended	**0.8931±0.08**	*0.7893±0.05*	0.7108±0.06	0.6743±0.10	0.4871±0.02
6%	Overall	Simple	**0.8924±0.02**	*0.7790±0.02*	0.7202±0.06	0.6810±0.04	0.4873±0.03
		Medium	**0.8922±0.01**	*0.7758±0.03*	0.7238±0.11	0.6800±0.03	0.4871±0.02
		Complex	**0.8921±0.03**	*0.7740±0.05*	0.7126±0.14	0.6698±0.02	0.4870±0.03
		Blended	**0.8920±0.05**	*0.7690±0.04*	0.7082±0.05	0.6690±0.02	0.4869±0.02
	UD	Simple	**0.8923±0.04**	*0.7680±0.06*	0.7100±0.07	0.6786±0.08	0.4868±0.03
		Medium	**0.8922±0.01**	*0.7588±0.03*	0.7098±0.08	0.6782±0.05	0.4864±0.04
		Complex	**0.8922±0.02**	*0.7560±0.06*	0.7090±0.01	0.6688±0.03	0.4863±0.01
		Blended	**0.8920±0.01**	*0.7520±0.10*	0.7084±0.03	0.6690±0.06	0.4862±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
		Medium	**0.8712±0.02**	*0.7590±0.08*	0.7130±0.05	0.6603±0.06	0.4864±0.02
		Complex	**0.8710±0.05**	*0.7441±0.04*	0.7014±0.06	0.6568±0.10	0.4864±0.03
		Blended	**0.8708±0.02**	*0.7380±0.05*	0.7010±0.07	0.6545±0.11	0.4860±0.02
	UD	Simple	**0.8713±0.04**	*0.7461±0.02*	0.7004±0.06	0.6600±0.12	0.4861±0.03

		Medium	**0.8709±0.06**	*0.7456±0.06*	0.6989±0.03	0.6555±0.02	0.4859±0.05
		Complex	**0.8703±0.03**	*0.7438±0.02*	0.6910±0.02	0.6515±0.06	0.4850±0.02
		Blended	**0.8732±0.02**	*0.7390±0.05*	0.6900±0.03	0.6508±0.04	0.4840±0.02

Table 3.12. Performance on RCL dataset using Precision

Noise combination			Precision				
			NoiseCleaner	HARF-80	HARF-70	HCleaner	CAIRAD
2%	Overall	Simple	**0.8701±0.02**	*0.8212±0.03*	0.7360±0.03	0.7240±0.04	0.4732±0.03
		Medium	**0.8702±0.02**	*0.8210±0.04*	0.7320±0.03	0.7184±0.04	0.4730±0.03
		Complex	**0.8700±0.06**	*0.8090±0.04*	0.7321±0.13	0.7187±0.08	0.4729±0.02
		Blended	**0.8700±0.05**	*0.8060±0.02*	0.7325±0.11	0.7185±0.11	0.4728±0.08
	UD	Simple	**0.8702±0.02**	*0.8221±0.03*	0.7311±0.04	0.7138±0.06	0.4727±0.05
		Medium	**0.8790±0.04**	*0.8145±0.04*	0.7216±0.04	0.7099±0.05	0.4726±0.03
		Complex	**0.8782±0.10**	*0.8126±0.01*	0.7241±0.11	0.7085±0.02	0.4724±0.01
		Blended	**0.8680±0.09**	*0.8121±0.02*	0.7138±0.10	0.7080±0.01	0.4725±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
		Medium	**0.8689±0.02**	*0.8085±0.04*	0.7245±0.06	0.6950±0.03	0.4721±0.02
		Complex	**0.8688±0.04**	*0.7989±0.03*	0.7205±0.10	0.6920±0.04	0.4722±0.01
		Blended	**0.8683±0.06**	*0.7988±0.02*	0.7165±0.11	0.6924±0.06	0.4719±0.03
	UD	Simple	**0.8688±0.03**	*0.7937±0.01*	0.7182±0.07	0.6960±0.07	0.4710±0.04
		Medium	**0.8676±0.04**	*0.7896±0.08*	0.7070±0.06	0.6887±0.06	0.4709±0.02
		Complex	**0.8673±0.02**	*0.7888±0.01*	0.7055±0.17	0.6813±0.06	0.4707±0.02
		Blended	**0.8670±0.01**	*0.7890±0.02*	0.7017±0.11	0.6810±0.04	0.4706±0.01
6%	Overall	Simple	**0.8682±0.05**	*0.7790±0.01*	0.7108±0.06	0.6834±0.03	0.4712±0.02
		Medium	**0.8681±0.04**	*0.7783±0.04*	0.7141±0.08	0.6809±0.10	0.4710±0.01

		Complex	**0.8679±0.08**	*0.7758±0.03*	0.7036±0.16	0.6800±0.06	0.4709±0.02
		Blended	**0.8678±0.09**	*0.7650±0.02*	0.7001±0.15	0.6740±0.07	0.4708±0.05
	UD	Simple	**0.8669±0.04**	*0.7640±0.03*	0.7104±0.06	0.6806±0.10	0.4707±0.06
		Medium	**0.8667±0.02**	*0.7620±0.02*	0.7101±0.07	0.6790±0.11	0.4705±0.03
		Complex	**0.8663±0.03**	*0.7409±0.04*	0.7080±0.14	0.6708±0.12	0.4700±0.03
		Blended	**0.8661±0.06**	*0.7401±0.08*	0.6998±0.12	0.6701±0.04	0.4701±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
		Medium	**0.8559±0.04**	*0.7402±0.04*	0.7023±0.04	0.6661±0.11	0.4706±0.02
		Complex	**0.8553±0.02**	*0.7401±0.06*	0.7015±0.05	0.6637±0.05	0.4706±0.01
		Blended	**0.8551±0.07**	*0.7400±0.07*	0.6987±0.12	0.6620±0.04	0.4705±0.03
	UD	Simple	**0.8643±0.02**	*0.7435±0.07*	0.7046±0.13	0.6623±0.04	0.4704±0.02
		Medium	**0.8631±0.04**	*0.7395±0.03*	0.7032±0.11	0.6610±0.06	0.4703±0.04
		Complex	**0.8524±0.02**	*0.7107±0.01*	0.7004±0.08	0.6585±0.03	0.4700±0.05
		Blended	**0.8521±0.03**	*0.7100±0.06*	0.6900±0.18	0.6502±0.10	0.4700±0.01

Table 3.13. Performance on MCI dataset using Precision

Noise combination			Precision				
			NoiseCleaner	HARF-80	HARF-70	HCleaner	CAIRAD
2%	Overall	Simple	**0.8731±0.01**	*0.7367±0.01*	0.6753±0.03	0.6305±0.06	0.4953±0.04
		Medium	**0.8728±0.05**	*0.7361±0.04*	0.6630±0.12	0.6304±0.12	0.4952±0.04
		Complex	**0.8725±0.02**	*0.7354±0.02*	0.6617±0.10	0.6301±0.18	0.4950±0.01
		Blended	**0.8716±0.03**	*0.7252±0.01*	0.6587±0.12	0.6300±0.03	0.4949±0.02
	UD	Simple	**0.8720±0.02**	*0.7248±0.04*	0.6710±0.13	0.6302±0.16	0.4939±0.03
		Medium	**0.8721±0.04**	*0.7263±0.05*	0.6508±0.09	0.6271±0.09	0.4938±0.02
		Complex	**0.8715±0.05**	*0.7182±0.03*	0.6500±0.08	0.6108±0.06	0.4836±0.01
		Blended	**0.8712±0.01**	*0.7101±0.01*	0.6489±0.12	0.6101±0.08	0.4835±0.02

4%	Overall	Simple	**0.8707±0.04**	*0.7285±0.07*	0.6587±0.10	0.6257±0.11	0.4835±0.01
		Medium	**0.8706±0.02**	*0.7281±0.02*	0.6589±0.06	0.6338±0.07	0.4833±0.02
		Complex	**0.8602±0.03**	*0.7187±0.03*	0.6567±0.06	0.6158±0.04	0.4830±0.02
		Blended	**0.8600±0.04**	*0.7068±0.04*	0.6561±0.05	0.6178±0.05	0.4831±0.02
	UD	Simple	**0.8725±0.03**	*0.7167±0.02*	0.6583±0.04	0.6280±0.06	0.4726±0.05
		Medium	**0.8628±0.06**	*0.7160±0.03*	0.6577±0.02	0.6228±0.11	0.4724±0.03
		Complex	**0.8625±0.04**	*0.7052±0.03*	0.6460±0.10	0.6191±0.10	0.4723±0.02
		Blended	**0.8614±0.03**	*0.7000±0.01*	0.6450±0.01	0.6187±0.02	0.4720±0.01
6%	Overall	Simple	**0.8722±0.08**	*0.7172±0.03*	0.6649±0.03	0.6303±0.05	0.4718±0.03
		Medium	**0.8720±0.04**	*0.7171±0.04*	0.6608±0.11	0.6312±0.01	0.4719±0.02
		Complex	**0.8618±0.03**	*0.7055±0.05*	0.6600±0.04	0.6176±0.02	0.4711±0.01
		Blended	**0.8616±0.04**	*0.6943±0.06*	0.6588±0.01	0.6074±0.03	0.4711±0.04
	UD	Simple	**0.8712±0.05**	*0.7100±0.03*	0.6610±0.06	0.6395±0.05	0.4710±0.02
		Medium	**0.8600±0.06**	*0.7087±0.03*	0.6584±0.06	0.6313±0.06	0.4709±0.02
		Complex	**0.8565±0.07**	*0.7047±0.02*	0.6556±0.04	0.6041±0.02	0.4703±0.03
		Blended	**0.8580±0.08**	*0.6998±0.04*	0.6497±0.02	0.6000±0.03	0.4702±0.01
8%	Overall	Simple	**0.8600±0.02**	*0.7141±0.04*	0.6581±0.07	0.6201±0.11	0.4706±0.04
		Medium	**0.8570±0.03**	*0.7140±0.05*	0.6587±0.03	0.6095±0.12	0.4705±0.01
		Complex	**0.8520±0.04**	*0.7110±0.03*	0.6571±0.04	0.6081±0.07	0.4703±0.02
		Blended	**0.8510±0.05**	*0.7090±0.02*	0.6467±0.10	0.5980±0.03	0.4702±0.01
	UD	Simple	**0.8558±0.06**	*0.7132±0.07*	0.6560±0.06	0.6013±0.02	0.4704±0.03
		Medium	**0.8546±0.07**	*0.7047±0.03*	0.6525±0.02	0.5965±0.10	0.4704±0.05
		Complex	**0.8531±0.02**	*0.7045±0.02*	0.6487±0.04	0.5905±0.06	0.4700±0.02
		Blended	**0.8526±0.03**	*0.7000±0.06*	0.6434±0.03	0.5901±0.06	0.4701±0.06

The noisy value detection accuracy of NoiseCleaner, HARF-80, HARF-70, and HCleaner on the four datasets in terms of $F_{0.5}$ for the 32 noisy combinations is presented in Tables 3.14 – 3.17. The $F_{0.5}$ of each table is calculated for the one set of data.

Table 3.14. Performance on RC dataset using $F_{0.5}$

Noise combination			$F_{0.5}$				
			NoiseCleaner	HARF-80	HARF-70	HCleaner	CAIRAD
2%	Overall	Simple	0.8721	0.7796	0.7340	0.6510	0.4720
		Medium	0.8716	0.7804	0.7354	0.6505	0.4719
		Complex	0.8711	0.7729	0.7304	0.6507	0.4701
		Blended	0.8707	0.7720	0.7293	0.6501	0.4694
	UD	Simple	0.8696	0.7794	0.7342	0.6504	0.4661
		Medium	0.8686	0.7797	0.7303	0.6478	0.4657
		Complex	0.8672	0.7752	0.7282	0.6496	0.4623
		Blended	0.8661	0.7740	0.7286	0.6494	0.4618
4%	Overall	Simple	0.8562	0.7712	0.7251	0.6499	0.4592
		Medium	0.8562	0.7692	0.7250	0.6482	0.4573
		Complex	0.8559	0.7695	0.7237	0.6408	0.4563
		Blended	0.8550	0.7677	0.7229	0.6399	0.4560
	UD	Simple	0.8628	0.7665	0.7250	0.6467	0.4494
		Medium	0.8550	0.7634	0.7223	0.6337	0.4487
		Complex	0.8564	0.7645	0.7199	0.6320	0.4480
		Blended	0.8558	0.7598	0.7186	0.6307	0.4447
6%	Overall	Simple	0.8555	0.7593	0.7214	0.6278	0.4348
		Medium	0.8540	0.7583	0.7205	0.6266	0.4344

		Complex	0.8462	0.7323	0.7186	0.6218	0.4313
		Blended	0.8454	0.7311	0.7170	0.6185	0.4312
	UD	Simple	0.8428	0.7383	0.7172	0.6170	0.4310
		Medium	0.8427	0.7304	0.7136	0.6159	0.4300
		Complex	0.8371	0.7295	0.7104	0.6149	0.4268
		Blended	0.8370	0.7214	0.6997	0.6118	0.4266
8%	Overall	Simple	0.8528	0.7245	0.6924	0.6087	0.4269
		Medium	0.8512	0.7233	0.6908	0.6051	0.4256
		Complex	0.8490	0.7202	0.6881	0.5958	0.4228
		Blended	0.8484	0.7160	0.6829	0.5944	0.4225
	UD	Simple	0.8480	0.7404	0.6788	0.5969	0.4220
		Medium	0.8468	0.7365	0.6751	0.5878	0.4180
		Complex	0.8454	0.7384	0.6531	0.5863	0.4141
		Blended	0.8450	0.7376	0.6659	0.5859	0.4249

Table 3.15. Performance on MII dataset using $F_{0.5}$

Noise combination			$F_{0.5}$				
			NoiseCleaner	HARF-80	HARF-70	HCleaner	CAIRAD
2%	Overall	Simple	0.8693	0.7688	0.7584	0.6504	0.4672
		Medium	0.8678	0.7684	0.7517	0.6568	0.4670
		Complex	0.8670	0.7688	0.7506	0.6572	0.4668
		Blended	0.8667	0.7687	0.7497	0.6563	0.4665
	UD	Simple	0.8664	0.7615	0.7507	0.6627	0.4665
		Medium	0.8659	0.7622	0.7557	0.6559	0.4663
		Complex	0.8649	0.7551	0.7383	0.6485	0.4657
		Blended	0.8643	0.7551	0.6938	0.6474	0.4654

4%	Overall	Simple	0.8655	0.7588	0.6970	0.6477	0.4576
		Medium	0.8571	0.7579	0.6968	0.6463	0.4564
		Complex	0.8568	0.7482	0.6922	0.6371	0.4562
		Blended	0.8557	0.7515	0.6821	0.6362	0.4540
	UD	Simple	0.8552	0.7516	0.6859	0.6426	0.4527
		Medium	0.8551	0.7507	0.6851	0.6408	0.4507
		Complex	0.8548	0.7488	0.6768	0.6351	0.4485
		Blended	0.8546	0.7509	0.6748	0.6307	0.4483
6%	Overall	Simple	0.8544	0.7413	0.6873	0.6311	0.4415
		Medium	0.8539	0.7389	0.6871	0.6277	0.4411
		Complex	0.8536	0.7376	0.6786	0.6203	0.4382
		Blended	0.8515	0.7337	0.6729	0.6168	0.4370
	UD	Simple	0.8518	0.7330	0.6739	0.6225	0.4365
		Medium	0.8515	0.7260	0.6738	0.6219	0.4359
		Complex	0.8515	0.7237	0.6721	0.6132	0.4354
		Blended	0.8507	0.7198	0.6716	0.6133	0.4346
8%	Overall	Simple	0.8453	0.7161	0.6682	0.5851	0.4364
		Medium	0.8373	0.7149	0.6637	0.5844	0.4339
		Complex	0.8369	0.7019	0.6539	0.5821	0.4335
		Blended	0.8366	0.6969	0.6535	0.5797	0.4326
	UD	Simple	0.8362	0.7020	0.6536	0.5797	0.4324
		Medium	0.8359	0.7008	0.6524	0.5768	0.4300
		Complex	0.8351	0.6980	0.6459	0.5700	0.4292
		Blended	0.8370	0.6936	0.6445	0.5697	0.4271

Table 3.16. Performance on RCL dataset using $F_{0.5}$

Noise combination			$F_{0.5}$				
			NoiseCleaner	HARF-80	HARF-70	HCleaner	CAIRAD
2%	Overall	Simple	0.8298	0.7581	0.6897	0.6329	0.4554
		Medium	0.8298	0.7565	0.6852	0.6290	0.4531
		Complex	0.8294	0.7479	0.6835	0.6285	0.4511
		Blended	0.8291	0.7450	0.6826	0.6261	0.4508
	UD	Simple	0.8289	0.7553	0.6818	0.6230	0.4480
		Medium	0.8352	0.7494	0.6732	0.6192	0.4480
		Complex	0.8342	0.7430	0.6747	0.6156	0.4451
		Blended	0.8267	0.7386	0.6646	0.6137	0.4448
4%	Overall	Simple	0.8286	0.7317	0.6703	0.6147	0.4489
		Medium	0.8276	0.7318	0.6700	0.6102	0.4427
		Complex	0.8274	0.7253	0.6658	0.6069	0.4400
		Blended	0.8269	0.7244	0.6628	0.6058	0.4386
	UD	Simple	0.8270	0.7207	0.6639	0.6073	0.4343
		Medium	0.8260	0.7165	0.6529	0.6028	0.4339
		Complex	0.8256	0.7149	0.6515	0.5979	0.4302
		Blended	0.8253	0.7146	0.6489	0.5976	0.4301
6%	Overall	Simple	0.8253	0.7047	0.6552	0.6025	0.4298
		Medium	0.8248	0.7038	0.6543	0.6004	0.4288
		Complex	0.8245	0.6960	0.6471	0.5994	0.4247
		Blended	0.8243	0.6870	0.6415	0.5952	0.4244
	UD	Simple	0.8235	0.6856	0.6482	0.5986	0.4226
		Medium	0.8232	0.6801	0.6468	0.5975	0.4225
		Complex	0.8223	0.6646	0.6354	0.5865	0.4252
		Blended	0.8216	0.6639	0.6299	0.5860	0.4227

8%	Overall	Simple	0.8211	0.6692	0.6127	0.5909	0.4226
		Medium	0.8134	0.6585	0.6096	0.5838	0.4219
		Complex	0.8125	0.6581	0.6053	0.5821	0.4218
		Blended	0.8120	0.6537	0.5988	0.5810	0.4180
	UD	Simple	0.8184	0.6573	0.6067	0.5811	0.4174
		Medium	0.8174	0.6512	0.5999	0.5794	0.4170
		Complex	0.8085	0.6319	0.5932	0.5775	0.4098
		Blended	0.8082	0.6284	0.5831	0.5721	0.4094

Table 3.17. Performance on MCI dataset using $F_{0.5}$

Noise combination			$F_{0.5}$				
			NoiseCleaner	HARF-80	HARF-70	HCleaner	CAIRAD
2%	Overall	Simple	0.8350	0.7012	0.6515	0.5905	0.4733
		Medium	0.8327	0.6985	0.6409	0.5887	0.4730
		Complex	0.8322	0.6962	0.6371	0.5871	0.4726
		Blended	0.8321	0.6877	0.6347	0.5867	0.4725
	UD	Simple	0.8313	0.6881	0.6400	0.5868	0.4718
		Medium	0.8309	0.6886	0.6251	0.5836	0.4714
		Complex	0.8282	0.6814	0.6222	0.5703	0.4637
		Blended	0.8277	0.6739	0.6210	0.5696	0.4632
4%	Overall	Simple	0.8197	0.6853	0.6227	0.5820	0.4619
		Medium	0.8185	0.6854	0.6225	0.5860	0.4636

		Complex	0.8109	0.6818	0.6207	0.5721	0.4610
		Blended	0.8081	0.6698	0.6199	0.5720	0.4611
	UD	Simple	0.8166	0.6790	0.6212	0.5789	0.4526
		Medium	0.8097	0.6725	0.6194	0.5750	0.4522
		Complex	0.8067	0.6642	0.6104	0.5722	0.4495
		Blended	0.8057	0.6578	0.6083	0.5690	0.4493
6%	Overall	Simple	0.8109	0.6692	0.6244	0.5787	0.4501
		Medium	0.8101	0.6669	0.6209	0.5743	0.4498
		Complex	0.8000	0.6575	0.6196	0.5648	0.4486
		Blended	0.7993	0.6475	0.6185	0.5564	0.4467
	UD	Simple	0.8062	0.6576	0.6179	0.5790	0.4466
		Medium	0.7982	0.6541	0.6154	0.5723	0.4458
		Complex	0.7940	0.6509	0.6101	0.5521	0.4448
		Blended	0.7943	0.6473	0.6057	0.5498	0.4444
8%	Overall	Simple	0.7959	0.6581	0.6130	0.5632	0.4434
		Medium	0.7916	0.6577	0.6131	0.5554	0.4431
		Complex	0.7875	0.6539	0.6104	0.5521	0.4401
		Blended	0.7872	0.6511	0.6031	0.5449	0.4400
	UD	Simple	0.7912	0.6537	0.6087	0.5464	0.4392
		Medium	0.7885	0.6480	0.6062	0.5427	0.4391
		Complex	0.7871	0.6478	0.6013	0.5353	0.4386
		Blended	0.7864	0.6448	0.5976	0.5334	0.4384

3.4.6 Execution time comparison

The average computation time in seconds for 160 datasets (32 combinations × 5 datasets per combination) for each traffic accident dataset is given in Table 3.18. The

configuration of the 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, and HARF-70. On the other hand, CAIRAD runs significantly faster compare to all the other approaches. We now analyse complexity for the NoiseCleaner algorithm. Let, x is the number of suspicious attributes values in the dataset. We find maximum n number of records from the S_s and S_d. The weighted similarity measure has to calculate for each x and the complexity is $O(nx)$.

Table 3.18. Average execution time of different algorithms (in seconds)

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

3.5 Conclusion

In this chapter, a brief summary of the algorithms proposed by researchers in the area of noisy values detections and corrections is first given. The literature review reveals that although there have been recent advances towards noisy values detection techniques and a number of algorithms are available, detecting and correcting incorrect values in categorical datasets is still a challenging problem, since any attempt to represent a categorical value numerically can introduce unwanted biases that negatively affect subsequent data analysis. A novel and effective noisy values identification and correction method, called NoiseCleaner, is proposed in this chapter for traffic accident data where the majority of attributes are categorical. There are two stages in NoiseCleaner, where noisy attribute values are first detected by using a novel P-measure

which computes the probability values indicating the likeliness of replacing the noisy attribute value with some alternative attribute values. Then, the S-measure, which measures the direct and transitive similarity between a noisy value and an alternative value, is used to identify the most similar alternative value to replace the noisy value with. Extensive experimental results demonstrated that it outperforms several existing noisy values identification methods on four real world traffic accident datasets.

Chapter 4: Conclusions and Future Research

This chapter concludes the thesis by summarising the novel contribution in this research and suggests some areas for future work. The objective of this research is to develop effective preprocessing algorithms for traffic accident data. Our research is focused on two pivotal tasks of data preprocessing, namely missing value imputation and data cleansing. To impute the missing values and detect the noisy data, detailed investigations are carried out on existing approaches to missing values imputation and noisy data detection in this research. From the investigations, it is found that although there is much published research on missing values imputation and noisy data detection, most of those methods are developed for data with numerical values. However, the traffic accident data we are concerned with in this research consists mostly of categorical values. Therefore, this research addresses the research gap of preprocessing categorical data by developing novel algorithms for missing value imputation and noisy value detection and correction.

In Chapter 2, we proposed a missing value imputation algorithm, called DSMI, which is able to deal with categorical data. Our algorithm takes into account the correlation between the attributes in a record, as well as the correlation between two records to find the most likely imputed values. Moreover, to model the variability in real data, our algorithm imputes the missing values by sampling from a list of potential imputed values based on their degree of affinity. Extensive experimental results have shown that DSMI significantly outperformed current state-of-the-art imputation algorithms.

In Chapter 3, we proposed a noisy value detection and correction called NoiseCleaner, which is able to deal with categorical data. NoiseCleaner detects noisy attributes values by using a novel *P-measure* which are probability values indicating the

likeliness of replacing the noisy attribute value with some alternative attribute values. Then, the *S-measure* which measures the direct and transitive similarity between a noisy value and an alternative value is used to identify the most similar alternative value to replace the noisy value with. The extensive experimental results and comparative studies presented in Chapter 3 indicate the effectiveness of the proposed algorithm.

4.1 Future Research

Although the proposed algorithms have demonstrated superior performance compared with existing algorithms, there is still scope for further research. Because of the availability of cheap sensors, massive amounts of data have been generated on a daily basis. Much of this data contain a mixture of numerical and categorical values. To handle the massive volumes of data, we need highly efficient data preprocessing algorithms. So far, our algorithms have only been tested on moderately large datasets. Investigating the scalability of our algorithms to big data is an important future research direction. In particular, it will be interesting to know how the various correlation measures used in our algorithms deal with data of very high dimension and the extent they are able to cope with the curse of dimensionality.

The ability to deal with mixed data types is also an important avenue for future research. Our proposed algorithms are specifically designed to handle categorical data. For numerical data of continuous value, one can often estimate or approximate the underlying density distribution. Exploiting this distribution is expected to improve the imputation or data cleansing performance. How to modify our algorithms so that they can handle both types of data will also be an interesting research direction to pursue in the future.

Twitter, Facebook, and LinkedIn have generated massive amount of data. However, missing values and noisy informations are common in these data. Future research can evaluate the effectiveness of the two preprocessing algorithms for these data. In medical record data, missing values can arise because people do not want to share sensitive

information. It would be useful to validate our preprocessing algorithms for this kind of dataset too.

Bibliography

[1] P. Miksovsky, K. Matousek, Z. Kouba, "Data pre-processing support for data mining", Proceeding of IEEE SMC Conference, Hammmet, Tunisia , pp. 1-8, 2002.

[2] R. Deb, A. W. C. Liew, E. Oh, "A correlation based imputation method for incomplete traffic accident data", Proceeding of PRICAI Conference, Springer LNAI, vol. 8862, pp. 905–912, Gold Coast, Australia, 2014.

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

[4] 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.

[5] "Traffic Safety Facts 2012 Data: United States Department of Transportation", https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/811888.

[6] "Australian Bureau of Statistics", http://www.abs.gov.au/ausstats/abs@.nsf/Lookup/by%20Subject/1301.0~2012~Main%20Features~Accidents,%20injuries%20and%20fatalities~189.

[7] "Data preprocess techniques for data mining", http://www.iasri.res.in/ebook/win_school_aa/notes/Data_Preprocessing.pdf.

[8] "Why data mining is important", http://booksite.elsevier.com/samplechapters/9781558609013/9781558609013.PDF.

[9] S. Gargett, L. B. Connelly, S. Nghiem, "Are we there yet? Australian road safety targets and road traffic crash fatalities", BMC Public Health, vol. 11, no. 270, pp. 323-336, 2011.

[10] Z. Zamani, M. Poumand, M.H. Saraee, "Application of data mining in traffic management: Case of city of Isfahan", Proceeding of ICECT Conference, Kuala Lumpur,

Malaysia, pp. 102-106, 2010.

[11] S. Zhang, C. Zhang, Q. Yang, "Data preparation for data mining", Applied Artificial Intelligence, vol. 17, no. 5-6, pp. 375-381, 2003.

[12] S. Kumar, D. Toshniwal, "A data mining framework to analyze road accident data", Journal of Big Data, vol. 2, no. 26, pp. 1-18,2015.

[13] E. Bayam, J. Liebowitz, W. Agresti, "Older drivers and accidents: A meta analysis and data mining application on traffic accident data", Expert Systems with Applications, vol. 29, no. 23, pp. 598-629,2005.

[14] J. L. Deng, "Control problems of grey system", Systems and Control Letters, vol. 1, no. 5, pp. 288-294, 1982.

[15] J. I. Maletic, A. Marcus, "Data cleansing: beyond integrity analysis", Proceeding of IQ Conference, USA, pp. 200-209, 2000.

[16] X. Zhu, X. Wu, Y. Yang, "Error detection and impact-sensitive instance ranking in noisy data sets", Proceeding of AAAI Conference, California, pp. 378-384, 2004.

[17] 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.

[18] R. Cheng, J. Chen, and X. Xie, "Cleaning uncertain data with quality guarantees", Proceeding of VLDB Endowment, vol. 1, No. 1, pp. 722-735, 2008.

[19] I. B. Aydilek, A. Arslan, "A hybrid method for imputation of missing values using optimized fuzzy c-means with support vector regression and a genetic algorithm", Information Sciences, vol. 233, pp.25–35, 2013.

[20] G. E. A. P. A. Batista, M. C. Monard, "An analysis of four missing data treatment methods for supervised learning", Journal of Applied Artificial Intelligence, vol. 17, no. 5-6, pp. 519-533, 2003.

[21] J. W. G.-Busse, M. Hu, "A comparison of several approaches to missing attribute values in data mining", Proceeding of RSCTC Conference, Springer LNAI, vol. 2005, pp. 378–385, Banff, Canada, 2001.

[22] K. O. Cheng, N. F. Law, W. C. Siu, "Iterative bicluster-based least square framework for estimation of missing values in microarray gene expression data", Pattern Recognition, vol. 45, no. 4, pp. 1281-1289, 2012.

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

[24] C. Gautam, V. Ravi, "Counter propagation auto-associative neural network based data imputation", Information Sciences, vol. 325, pp. 288-294, 2015.

[25] 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.

[26] 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.

[27] C.-C. Liu, D.-Q. Dai, H. Yan, "The theoretic framework for local weighted approximation for microarray missing value estimation", Pattern Recognition, vol. 43, no. 8, pp. 2993-3002, 2010.

[28] E.-L. S.-Ramírez, R. P.–Mejías, M. L.–Colllo, M.-D. C.-D.-L. Vela, "Missing value imputation on missing completely at random data using multilayer perceptions", Neural Networks, vol. 24, no. 1, pp. 121-129, 2011.

[29] M. G. Rahman, M. Z. Islam, "FIMUS: A framework for imputing missing values using co-appearance, correlation and similarity analysis", Knowledge-Based Systems, vol. 56, pp. 311-327, 2014.

[30] M. G. Rahman, M. Z. Islam, "*k*-DMI: A novel method for missing values imputation using two levels of horizontal partitioning in a data set", Proceeding of ADMA Conference, Hangzhou, China, pp. 250-263, 2013.

[31] M. G. Rahman, M. Z. Islam, "Missing value imputation using decision trees and decision forests by splitting and merging records: Two novel techniques", Knowledge-Based Systems, vol. 53, pp. 51-65, 2013.

[32] M. G. Rahman, M. Z. Islam, "Missing value imputation using a fuzzy clustering-based EM approach", Knowledge and Information Systems, vol. 46, no. 2, pp. 389-422, 2016.

[33] T. Schneider, "Analysis of incomplete climate data: estimation of mean values and covariance matrices and imputation of missing values", Journal of Climate, vol. 14, no. 5,

pp. 853-871, 2001.

[34] C.-F. Tsai, F.-Y. Chang, "Combining instance selection for better missing value imputation", The Journal of Systems and Software, vol. 122, pp. 63-71, 2016.

[35] V. Ravi, M. Krishna, "A new online data imputation method based on general regression auto associative neural network", Neurocomputing, vol. 138, pp. 207–212, 2014.

[36] K. Muralidhar, R. Parsa, R. Sarathy, "A general additive data perturbation method for database security", Management Science, vol. 45, no. 10, pp. 1399 - 1415, 1999.

[37] P. J. Rathouz, "Missing data: weighting and imputation", Encyclopaedia of Health Economics, pp. 292–298, 2014.

[38] P. C. Austin, M. D. Escobar, "Bayesian modeling of missing data in clinical research", Computational Statistics & Data Analysis, vol. 49, no. 3, pp. 821-836, 2005.

[39] J. Tang, G. Zhang, Y. Wang, H. Wang, F. Liu, "A hybrid approach to integrate fuzzy C-means based imputation method with genetic algorithm for missing traffic volume data estimation", Transportation Research Part C: Emerging Technologies, vol. 51, pp. 29–40, 2015.

[40] L. L. Doove, S. V. Buuren, E. Dusseldorp, "Recursive partitioning for missing data imputation in the presence of interaction effects", Computational Statistics & Data Analysis, vol. 72, pp. 92–104, 2013.

[41] M. Duma, T. Marwala, B. Twala, F. Nelwamondo, "Partial imputation of unseen records to improve classification using a hybrid multi-layered artificial immune system and genetic algorithm", Applied Soft Computing, vol. 13, no. 12, pp. 4461–4480, 2013.

[42] C. Gautam, V. Ravi, "Data imputation via evolutionary computation", clustering and a neural network, Neurocomputing, vol. 56, pp.134–142, 2015.

[43] F. V. Nelwamondo, D. Golding, T. Marawala, "A dynamic programming approach to missing data estimation using neural networks", Information Sciences, vol. 237, 49–58, 2013.

[44] K. J. Nishanth, V. Ravi, "A computational intelligence based on line data imputation method: an application for banking", Journal of Information Processing Systems, vol. 9, no. 4, pp. 633–650, 2013.

[45] P. M. V. Rancoita, M. Zaffalon, E. Zucca, F. Bertoni, C. P. Campos, "Bayesian network

data imputation with application to survival tree analysis", Computational Statistics & Data Analysis, vol. 98, pp. 89–97, 2015.

[46] B. N. Howie, P. Donnelly, J. Marchini, "A flexible and accurate genotype imputation method for the next generation of genome-wide association studies", PLoS Genetics, vol. 5, no. 6, pp. e1000529, 2009.

[47] Q. Song, M. Shepperd, "A new imputation method for small software project datasets", Journal of Systems and Software, vol. 80, no. 1, pp. 51–62, 2007.

[48] J. K. Dixon, "Pattern recognition with partly missing data", IEEE Transactions on Systems, Man, and Cybernetics, vol. 9, no. 10, pp. 617-621, 1979.

[49] P. Jhosson, C. Wohlin, "An evaluation of k-nearest neighbour imputation using Likert data", 10th International Symposium on Software Metrics, pp. 108-118, 2004.

[50] D. J. Stekhoven, P. Bühlmann, "MissForest—non-parametric missing value imputation for mixed-type data", Bioinformatics, vol. 28, no. 1, pp. 112-118, 2012.

[51] S. Erdogan, I. Yilmaz, T. Baybura, M. Gullu, "Geographical information systems aided traffic accident analysis system case study: city of Afyonkarahisar", Accident Analysis & Prevention, vol. 40, no. 1, pp. 174-181, 2008.

[52] A. Hossain, M. Chattopadhyay, S. Chattopadhyay, S. Bose, C. Das, "A bicluster-based sequential interpolation imputation method for estimation of missing values in microarray gene expression data", Current Bioinformatics, vol. 12, no. 2, pp. 118-130, 2017.

[53] J. Y. Nancya, N. H. Khannaa, K. Arputharajb, "Imputing missing values in unevenly spaced clinical time series data to build an effective temporal classification framework", vol. 112, pp. 63–79, 2017.

[54] S. K. Pati, A. K. Das, "Missing value estimation for microarray data through cluster analysis", Knowledge and Information Systems, pp. 1-42, 2017.

[55] A. P. Dempster, N. M. Laird, D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm", Journal of the Royal Statistical Society. Series B (Methodological), vol. 39, no. 1, pp. 1-38, 1977.

[56] H. P. Giggins, L. Brankovic, "VICUS – a noise addition technique for categorical data", Proceeding of AusDM Conference, Sydney, Australia, pp. 139–148, 2012.

[57] 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.

[58] D. R. Wilson, T. R. Martinez, "Reduction Techniques for Instance-Based Learning Algorithms", Machine Learning, vol. 38, no. 3, pp. 257-286, 2000.

[59] M. Lee, W. Pedrycz, "The fuzzy C-means algorithm with fuzzy P-mode prototypes for clustering objects having mixed features", Fuzzy Sets and Systems, vol. 160, no. 24, pp. 3590-3600, 2009.

[60] J. R. Quinlan, "C4.5: programs for machine learning", Morgan Kaufmann Publishers, San Mateo, California, USA, 1993.

[61] J. R. Quinlan, "Improved use of continuous attributes in C4.5", Journal of Artificial Intelligence Research, vol. 4, pp. 77-90, 1996.

[62] P.-N. Tan, V. Kumar, "Interestingness measures for association patterns: a perspective", Proceeding of KDD workshop on post-processing in Machine Learning and Data Mining, pp. 254-260, Boston, USA, 2000.

[63] P.-N. Tan, V. Kumar, J. Srivastava, "Selecting the right objective measure for association analysis", Information Systems, vol. 29, No. 4, pp. 293–313, 2004.

[64] L. Geng, H. J. Hamilton, "Interestingness measures for data mining: A Survey", ACM Computing Surveys, vol. 38, No. 3, pp. 1-32, 2006.

[65] T. Brijs, G. Swinnen, K. Vanhoof, G. Wets, "Using association rules for product assortment decisions: a case study", Proceeding of KDD conference, pp. 254-260, San Diego, California, USA, 1999.

[66] C. Silverstein, S. Brin, R. Motwari, "Beyond market baskets: generalizing association rules to dependence rules", Data Mining and Knowledge Discovery, vol. 2, No. 1, pp. 39-68, 1998.

[67] T. Chai, R. R. Draxler, "Root mean square error (RMSE) or mean absolute error (MAE)? – arguments against avoiding RMSE in the literature", Geoscientific Model Development, vol. 7, pp. 1247-1250, 2014.

[68] 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.

[69] 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.

[70] 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.

[71] 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.

[72] B. Sluban, D. Gamberger, N. Lavraˇc, "Ensemble-based noise detection: noise ranking and visual performance evaluation", Data Mining and Knowledge Discovery, vol. 28, no. 2, pp. 265-303, 2013.

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

[74] 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.

[75] "Queensland Government data repository",

https://data.qld.gov.au/dataset/crash-data-from-queensland-roads.

[76] "New York's open data portal of Motor Vehicle crash – case information: 2011 and Motor Vehicle crash - individual information: 2011",

https://data.ny.gov/browse?tags=crash&utf8=%E2%9C%93.

[77] "Accident Investigation Basics",

http://www.crashinvestigation.net.au/bob_lomastroDefnRootCauseAnalysispage10.pdf.

[78] "Large Truck Crash Causation Study File 1 and 2", explore.data.gov.

[79] "Traffic Accidents datasets of Denver County", http://data.opencolorado.org/dataset/city-and-county-of-denver-traffic-accidents.

[80] J. C. Bezdek, R. Ehrlich, W. Full, "FCM: The fuzzy c-means clustering algorithm", Computer and Geosciences, vol. 10, no. 2-3, pp. 191-203, 1984.

[81] J. Han, M. Kamber, "Data mining: concepts and techniques. The Morgan Kaufmann series in data management systems", 2000.

[82] 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.

[83] R. Deb, A. W. C. Liew, "Missing value imputation for the analysis of incomplete traffic accident data", Proceeding of ICMLC Conference, Springer CCIS 481, pp. 275–286, Guangzhou, China, 2014.

[84] X. Zhu, S. Zhang, Z. Jin, Z. Zhang, Z. Xu, "Missing value estimation for mixed-attribute data sets", IEEE Transactions on Knowledge and Data Engineering, vol. 23, no. 1, pp. 110-121, 2011.

[85] MATLAB version 7.10.0. Natick, Massachusetts: The Math Works Inc., 2010.

[86] N. Lavesson, S. Axelsson, "Similarity assessment for removal of noisy end user license agreements", Knowledge Information System, vol. 32, no. 1, pp. 167-189, 2011.

[87] I. Tomek, "An experiment with the nearest neighbor rule", IEEE Transactions on Information Theory, vol. 6, pp. 448–452, 1976.

[88] L. I. Rudin, S. Osher, E. Fatemi, "Nonlinear total variation based noise removal algorithms", Physica D: Nonlinear Phenomena, vol. 60, no. 1-4, pp. 259-268, 1992.

[89] R. Garnett, T. Huegerich, C. Chui, W. He, "A universal noise removal algorithm with an impulse detector", IEEE transactions on image processing, vol. 14, no. 11, pp. 1747-1754, 2005.

[90] R. J. Hathaway, J. C. Bezdek, "Fuzzy c-means clustering of incomplete data", IEEE transactions on Systems, Man, and Cybernetics-Part B: Cybernetics, vol. 31, no. 5, pp. 735-744, 2001.