APPLICATION OF DATA DUPLICATE DETECTION AND FUSION TO IMPROVE DATA QUALITY

Author

FATEH-UR-REHMAN

00000172057

MS-16(CSE)

Supervisor

DR. MUHAMMAD ABBAS

DEPARTMENT OF COMPUTER & SOFTWARE ENGINEERING

COLLEGE OF ELECTRICAL & MECHANICAL ENGINEERING

NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY

ISLAMABAD

OCTOBER, 2018

APPLICATION OF DATA DUPLICATE DETECTION AND FUSION TO IMPROVE DATA QUALITY

Author

FATEH UR REHMAN

00000172057

MS-16(CSE)

A thesis submitted in partial fulfilment of the requirements for the degree of MS Computer Software Engineering

Thesis Supervisor

DR. MUHAMMAD ABBAS

Thesis Supervisor's

Signature: _____

DEPARTMENT OF COMPUTER & SOFTWARE ENGINEERING

COLLEGE OF ELECTRICAL & MECHANICAL ENGINEERING

NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY

ISLAMABAD

OCTOBER, 2018

AND THEY CAN'T ENCOMPASS ANYTHING FROM HIS KNOWLEDGE, BUT TO EXTEND HE WILLS [2:255]

DECLARATION

I certify that this research work titled *"Application of Data Duplicate Detection and Fusion to Improve Data Quality"* is my own work. The work has not been presented elsewhere for assessment. The material that has been used from other sources it has been properly acknowledged/referred.

FATEH UR REHMAN
00000172057
MS-16(CSE)

LANGUAGE CORRECTNESS CERTIFICATE

This thesis has been read by an English expert and is free of typing, syntax, semantic, grammatical and spelling mistakes. The thesis is also according to the format given by the university.

_____

FATEH UR REHMAN
00000172057
MS-16(CSE)

_____

DR. MUHAMMAD ABBAS

COPYRIGHT STATEMENT

ACKNOWLEDGEMENTS

First of all, I am extremely thankful to Almighty Allah for giving me courage and strength to complete this challenging task and to compete with the international research community. I am also grateful to my family members, especially my parents, brother and sisters who have encouraged and supported me through their prayers that have always been with me.

I am highly obliged to Dr Muhammad Abbas for his valued suggestions and uninterrupted guidance throughout my research work. I am highly thankful to the committee members for their support and help throughout the research work.

I am also grateful to all of my instructors who have been overseeing me throughout my coursework and have contributed to my knowledge. Their guidance, training and knowledge helped me a lot to carry out this research work.

FATEH UR REHMAN

To my parents, sisters, brother and teachers those fabulous support and assistance led me to this magnificent achievement

ABSTRACT

In the current digital era, data generation sources are producing abundant data volumes related to each field of life. Volumes of data are expanding every second because sources of data creation are also expanding. Machines (i.e. sensors, devices) and humans (i.e. filled forms, saved data) are the main sources of data created in this digital era. Machines record each and every relevant event in their recording medium by combining the recorded inputs. Occasionally, machines receive only a few inputs and save them by keeping the missing input values as null. Sometimes an identical event is logged twice as duplicate data by the single or multiple machines in the same recording medium. Duplicate data and missing values issues are also found in human-generated data mostly due to human error. Missing values and duplicates affect the quality of data and biases the data mining results. An efficient Data Cleansing System (DCS) is developed during the research to improve the quality of data by filling the missing values and removing the duplicate records from the dataset. Similarity-based (SimFiller) and duplicate detection based (DuDeFiller) missing values filling algorithms are developed and integrated into the system to fill the missing values of a record by taking the missing value replacement from its most similar or duplicate record. Duplicate detection based data fusion algorithm (DuDeFuse) is also developed and integrated into the system to merge the duplicate records based on the maximum similarity of the attribute's value or the maximum occurrence of the attribute's value. Two data sets from the UCI machine learning repository are cleaned through the system to improve their data quality. The selected datasets are also cleaned through the five others exiting missing values filling algorithms for the purpose of comparison. Five classifiers including "Naive Bayes", "Decision Tree", "Random Forest", "Deep Learning", and "Logistic Regression" are selected to check the classification accuracy and The f-measure of the cleaned datasets. Average classification accuracy of both datasets is increased up to 3.00% after cleansing the datasets with the developed system. The f-measure of the datasets is also increased up to 3.26% after cleansing them with the developed system as compared to the data cleansing performed with the other algorithms. The developed system can be extended to resolve the other inconsistencies in the datasets and can also be evaluated for the other datasets.

Keywords: Data preprocessing, Data analysis, Data engineering, Data cleansing, Duplicate detection, Duplicates, Missing values, Null values.

TABLE OF CONTENTS

TABLE OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

CSV file	Comma Separated Values file
DASD	Dresses Attribute Sales Dataset
DCS	Duplicate Cleansing System
DCSNM	Duplicate Count SNM
DDS	Duplicate Detection Subsystem
DFS	Duplicate Fusion Subsystem
DFS-MV	Duplicate Fusion Subsystem for Missing Values
DuDe	Duplicate Detection Toolkit
DuDeFiller	Duplicate Detection based missing values Filling algorithm
DuDeFuse	Duplicate Detection based data Fusion algorithm
K-NN	K-Nearest Neighbors
MI	Multiple Imputations
ML	Machine Learning
POS	Post-Operation Subsystem
PPS	Pre-Processing Subsystem
RM	RapidMiner Studio
RMVO	Replace Missing Value Operator
SI	Single Imputations
SimFiller	Similarity-based missing values Filling algorithm
SIP	Solo Iterative Process
SNM	Sorted Neighborhood Method

RESEARCH CONTRIBUTION / PUBLICATION

Following research articles has been produced by this research:

- SimFiller: Similarity-Based Missing Values Filling Algorithm, Fateh R. et al. (2018), Published by IEEE in 13th ICDIM 2018 of Berlin, Germany.
- DuDeFiller: Duplicate detection based missing values filling algorithm, Fateh R. et al. (2018) submitted in Journal.

Chapter 1

INTRODUCTION

This chapter introduces the research work carried out and presented in this thesis. It also describes the problem statement, proposed solution for the recognized problem and the motivational factors to carry out this research.

1.1. Overview

During this current digital era, manual files systems are replaced with the computers, machines and database servers and other than the recording of data within the recording medium, automatic machines (sensing and recording devices) are also producing data volumes related to each field of life. A number of data generation sources are growing on a daily basis and continuously multiplying the data volumes.

Machines (i.e. sensors, devices) and humans (i.e. filled forms, saved data) are the key sources of data created in this digital era. Digital systems administrators are authorizing devices to record each and every minute and relevant instance in their storage medium. Data generation sources attempt to record each and every instance of received data without checking its quality with respect to completeness and duplication.

A record in the database is created by combining the values of different attributes from similar or different sources. Sometimes, machines receive only a few inputs and save them by keeping the missing input values as null. Occasionally, an identical event is logged twice as duplicate data by the single or multiple machines in the same recording medium. Duplicate data and missing values issues are caused due to machine error like sensor error and these issues also exist in human-generated data mostly due to human error, typo errors or other input device related errors.

Data anomalies are divided into two categories i.e. systematic inconsistencies and non-systematic inconsistencies [1-3]. Systematic irregularities or non-random inconsistencies have some common error pattern and caused due to device error or device limitation to record specific types of incidents. Non-systematic irregularities or random inconsistencies are occurred at a certain time only due to some rare event. Systematic and non-systematic problems cause irregularities of missing values and duplicate record.

Missing values and duplicate record issues are also produced if two or more datasets are merged to create a single dataset. Data irregularity problems are found in all of the data sets belonging to every domain of life including customer associated databases, commercial information databases, countrywide census databases, patient illness databases, virus databases, campaign management datasets, nationwide security databases and bibliography datasets.

Missing values and duplicate data issues severely upset the quality of data and create problems for successful data mining process. The dataset inconsistencies lead to unreliable and biases data mining results [1] and these irregularities particularly affect the process of supervised learning [4]. Therefore, missing values and duplicates records need to be cleaned prior to making decisions, predictive analysis and forecasting.

Most of the researchers focus to fill the missing values with any non-null, valid and suitable replacement value [5, 6]. Duplicate cleansing is a process of duplicate record detection and merging. Duplicate detection discovers groups of records which belong to the same real-world entity [7]. Once duplicate records are detected, they need to be merged to form a single updated record. This new record will replace all its duplicates. Consequently, a clean dataset is achieved for any further processing.

Manually fixing the dataset inconsistencies is completely unmanageable due to the size of the dataset and because of the complications connected with inconsistencies. Data preprocessing techniques [8, 9] were introduced to fix the irregularities of the datasets, including missing values and duplicate records, by filling missing values with their appropriate substitutes and by eliminating or replacing the duplicate records.

Several techniques were suggested by the data researchers to fix the irregularities of the datasets including removal of the null and duplicate values records. Every proposed technique was either devised for a particular dataset or developed and tested for a specific classifier. Duplicate detection based data cleansing is also one of the methods used to fix the data inconsistencies and is based on duplicate record discovery. Newcombe et al. [10] primarily considered duplicate detection for data cleansing and researchers later worked on similar concepts with different names like data cleansing, record reconciliation, object identification, entity resolution, field matching, merge purge, and data matching.

Java code library of all eminent duplicate detection algorithms is also developed by researchers with the name of Duplicate Detection Toolkit (DuDe) [7]. The architecture of the Duplicate Detection Toolkit (DuDe) is shown in Fig. 1 which explains the functionality of DuDe and indicates that DuDe extracts inconsistent data from the data source, considers it as input, preprocesses it, applies partitioning algorithm on it, compares partitions, post processes it and generate an output of cleaned data.

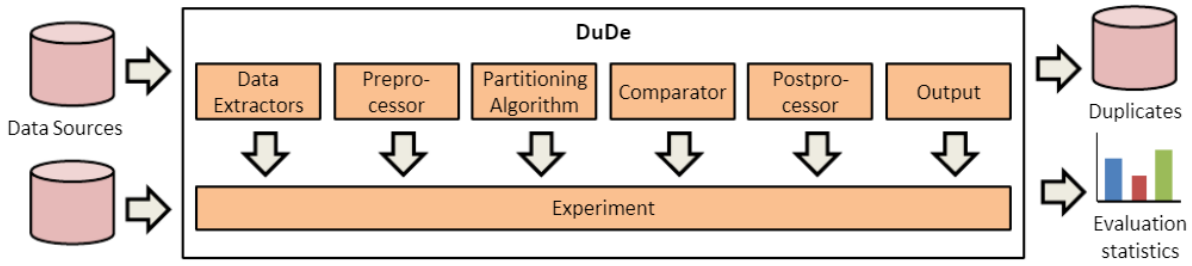

Fig. 1. The Architecture of the Duplicate Detection Toolkit

This research is emphasized to solve the two types of dataset irregularities including filling the missing values with the proper substitute and cleansing and fusing the duplicates.

1.2. Motivation

In this digital era, data quality problems including the irregularities of missing values and duplicate data are increasing due to the increase in heterogeneous data generation sources [4, 11]. Data preprocessing algorithms are least improved as compared to the improvements carried out in other data science fields. Most of the data scientists are concentrating to improve the correctness of data mining operators and algorithms.

Data mining operators generate the data mining results from the given input data without considering the quality of input data and it causes the unfairness in data mining results [2, 3]. Duplicate Detection Toolkit (DuDe) [7] was developed by researchers containing a wide range of duplicate detection algorithms. An opportunity was found to extend the DuDe toolkit for filling the missing values and solving the duplicate data problems.

1.3. Problem Statement

Data quality problems are increasing on daily basis due to the increase in data generation sources. There are only a few solutions available to fix the data inconsistencies related to missing values and to fuse the duplicate data. Moreover, the previously developed solutions

were developed and tested for a specific dataset or operator. This research is intended to build a proficient and innovative technique to improve the quality of any dataset by filling the missing values through suitable substitution and by eliminating the duplicate records from the dataset without disturbing the diversity and classification accuracy of the entire dataset.

1.4. Proposed Solution

A data quality improvement system, named as ***Data Cleansing System (DCS)***, is developed during this research to fill the missing or null values of a dataset and to merge the duplicate records of the dataset. The developed system is based on duplicate detection mechanism which extracts the duplicate records in the form of pairs named as duplicate pairs. These duplicate pairs have some similarities (duplication) among them and this similarity is greater than the similarity of duplicate records. The extracted duplicate pairs are utilized to take the duplicate merging decisions by the system. Moreover, the developed system also utilizes these duplicates pairs to fill the missing values within the pairs.

1.5. Publication

Following research article has been produced by this research:

- SimFiller: Similarity-Based Missing Values Filling Algorithm, Fateh R. et al. (2018), Published by IEEE in 13th ICDIM 2018 of Berlin, Germany.

1.6. Organization of the thesis

This chapter introduces the problem and the work carried out and presented in this thesis. It defines the problem statement, proposed solution for the recognized problem and the motivational factors to carry out this research.

Chapter 2 is related to literature review which discusses the literature of data cleansing and preprocessing related to missing values and duplicate cleansing. This chapter is divided into two parts to explain the existing solutions of two different anomalies of datasets. The first part deals with the literature review to extract the algorithms and tools related to filling the missing values. The second part deals with the review of the literature to discover the tools and algorithms associated with the cleansing of duplicate records.

Chapter 3 discusses the solution developed as a result of research on data quality issues. The developed system named Data Cleansing System (DCS) attempts to solve the dataset quality problems for missing and duplicate values are presented in this chapter. The developed system is based on duplicate detection mechanism during which record having some similarities (duplication) with each other are extracted in the form of pairs named as duplicate pairs.

Chapter 4 comprises the description of experiments setup used to get the results for the proposed algorithms in the research. The setup was created to validate the duplicate detection results, missing values filling results and duplicate fusion results for the correctness and endorsement of the contribution made by this research in the body of knowledge.

Experiments are conducted after completing the evaluation setup and the achieved results are presented and discussed in Chapter 5. This chapter comprises the duplicate detection results, missing values filling results and duplicate fusion results of experiments carried out for the proposed system. Results show the correctness and validation of the contribution made by the research in the body of knowledge.

Chapter 6 discusses the outcomes and the impact of the proposed study and concludes the overall benefit and result from this study. This chapter also defines the future directions and advises the prospect research from the presented idea in the earlier chapters.

Chapter 2

LITERATURE REVIEW

This chapter is related to the review of the literature and discusses the existing work on data cleansing and preprocessing to fix the issues of data inconsistencies. Missing values replacement algorithms and duplicate records removal algorithms presented by previous researchers are also discussed in this chapter.

2.1. Considered Data Inconsistencies

The dataset inconsistencies are of different types causing the quality problems for data. This research considers two important dataset inconsistencies given below:

- Missing Values
- Duplicate Records

This research discusses the existing approaches presented by previous researchers to deal with the dataset inconsistencies given above. This research also discusses the problems in existing approaches and tries to resolve the above-mentioned inconsistencies in a better and efficient way without affecting the dataset variability.

The literature review is divided into two parts to explain the existing solutions of two different anomalies of datasets. The first part deals with the literature review to extract the algorithms and tools related to filling the missing values. The second part deals with the review of the literature to discover the tools and algorithms associated with the detection and cleansing of duplicate records.

2.2. Missing Values

This section deals with the techniques for filling the missing values of records in a dataset. Missing values are produced due to the machine and human error caused during the generation of the dataset. Machines (i.e. sensors, devices) and humans (i.e. filled forms, saved data) are the key sources of data created in this digital era. Digital systems administrators are authorizing devices to record each and every minute and relevant instance in their storage medium. Data

generation sources attempt to record each and every instance of received data without checking its quality with respect to completeness.

Missing values severely upset the quality of data and create problems for successful data mining process. The dataset inconsistencies lead to unreliable data mining results [1] and these irregularities particularly affect the process of supervised learning [4, 12]. Therefore, missing values need to be cleaned prior to decisions making, predictive analysis and forecasting. Most of the researchers focus to fill the missing values with any non-null, valid and suitable replacement value [5, 6].

2.3. Types of Missing Values

Dataset missing values are classified into two different types based on the similarities and the differences among them. Systematic and non-systematic problems cause irregularities of missing values.

Two different types of missing values are given below:

- Systematic Irregularities
- Non-Systematic Irregularities

1) Systematic Irregularities

These irregularities are also named as non-random inconsistencies because these irregularities cover the missing values those are caused due to systematic or non-random error. Systematic irregularities or non-random inconsistencies have some common error pattern and caused due to device error or device limitation to record specific types of incidents [1-3].

2) Non-Systematic Irregularities

These irregularities are also named as random inconsistencies because these irregularities cover the missing values those are caused due to non-systematic or random error. Non-systematic irregularities or random inconsistencies are occurred at a certain time only due to some rare or special event which happens occasionally [1-3].

2.4. Missing Values Filling Techniques

Data scientists proposed their techniques to treat the missing and null values of the datasets. These methods are classified into different categories based on the operations performed during them.

Categorization of the proposed techniques are given below:

- Ignore, Delete, Discard
- Single Imputation (SI)
- Multiple Imputations (MI)
- Machine Learning (ML)

1) Ignore, Delete, Discard

The simpler method (for dealing with the missing values) is to delete, discard or ignore records containing the missing values while performing any data analysis on inconsistent data [13-15]. Records deletion can negatively influence the process of data mining because there is a chance to delete the records containing the most important data [16].

It is not a suitable method when datasets containing missing values due to systematic or non-random error and have a large number of records with missing values. Bulk removal of records can completely eliminate the specific class label from the process of data analysis and can adversely affect the data mining results [12].

This is an appropriate method when datasets containing missing values due to random or non-systematic error and have less number of records with missing values. It is a more beneficial choice to delete a small number of records containing the majority of its attributes having null value [13].

2) Single Imputations (SI)

This method specifies a particular value as a replacement of all missing values within a dataset. A single replacement value can be defined for all the missing values of all attributes (one value for all dataset) or multiple replacements can be defined for all the missing values of each attribute (one value for each attribute) [17]. Both types of single

value replacement (database level or attributes level) are performed on a single copy of the input dataset under consideration [2, 17].

This technique disturbs the diversity of a dataset and biases the accuracy of a classifier when the analysis is performed on it because single value replacement either on database level or on attribute level is neither an appropriate replacement nor a suitable representation of all the existing values of an attribute [6, 17, 18].

i. Mean

This technique specifies an average or means value as a replacement of all missing values within a dataset. A single mean value is calculated for each attribute of the dataset and in this way multiple replacements are defined for the missing values of each attribute (single average value for each attribute) [5, 6].

This method considers the attributes independence which is not suitable for each and every dataset. It also disturbs the variability of a dataset and biases the accuracy of classifiers when data analysis is performed. A single value for all the missing values can't indicate the variability of the dataset and can affect the classification accuracy [12, 17, 19].

ii. Hot and Cold Deck

This method selects the replacement value within the completed records and replaces it against the missing values of the considered dataset. Multiple values are defined as replacement of the missing values by creating the multiple clusters. One value from each cluster is selected for the records within that cluster. Hot deck and cold deck are two variations of this methodology for filling the missing values based on the selection of completed values as replacements [6, 17, 18].

Hot deck fills the missing value by getting the relevant replacement from the same single dataset which means replacing and replaced records are found on the same dataset [6, 17]. Cold deck fills the missing value by getting the relevant replacement from the different dataset which means replacing and replaced records are found on the different dataset [18].

This technique disturbs the diversity of a dataset and biases the accuracy of a classifier when the analysis is performed on it because randomly selected single value replacement either from the same dataset or from the different dataset is neither an appropriate replacement nor a suitable representation of all the existing values of an attribute [2, 17].

iii. Most Likely

This methodology considers each attribute (one by one) and calculates their missing values' replacements from the non-null values. Maximum occurred non-null attribute's value is selected as replacement of missing values for that attribute [5, 6, 19].

Missing value replacement can also be selected by calculating the probability of each value. This is one variation of this method in which the missing value is calculated from the subset of data having the same class label [5, 6].

This method considers the independence of attributes which is not suitable for each and every dataset. It also disturbs the variability of a dataset and biases the accuracy of classifiers when data mining is performed. A single value either for the complete dataset or for a specific subset of data can affect the variety of dataset and can disturb the classification accuracy [12, 17, 19].

iv. Not Applicable

This is a special method to treat the missing and null values of the dataset by substituting them with a predefined tag similar to "Not Applicable", "N/A", "Undefined" or "Not Available" [4]. This method is only suitable to recognize the missing values within a dataset and shows an adverse effect on the accuracy of the dataset. It is a suitable methodology when some dataset can't be filled for its missing values since some values remain not applicable in some instances and applicable for some instances[6].

This method is not suitable for each and every dataset and disturbs the variability of a dataset and biases the accuracy of classifier when data mining is performed. A single value for all the missing values can't indicate the variability of example and can affect the classification accuracy of the dataset [17, 19].

3) Multiple Imputations (MI)

This is an arithmetical method which calculates the several replacements for missing values and accordingly creates several copies of a dataset filled with the respected replacement values. Total x number of database copies are created for x number of arithmetically calculated replacements for missing values. Each copy of dataset is passed through the numerical investigation, arithmetical examination and "Exception Maximization" calculates the value of probable replacement similar to the Bayesian method and final missing value replacement is selected [17, 18].

This process reflects the individuality of attributes which is not appropriate for all types of datasets. It also disturbs the variability of a dataset and biases the accuracy of classifiers when data analysis is performed. Specific values either from the complete dataset or from a specific subset of data can affect the variety of dataset and can disturb the classification accuracy [17, 19].

4) Machine Learning (ML)

Different machine learning techniques are presented by investigators to fill the missing values of the dataset with an appropriate substitute. Researchers presented methods to fill the missing values by creating the subsets of dataset like subgroups having a similar class label or clusters of similar records [13, 15]. This methodology considers a missing value record as test data and trains the model to find the missing value attribute as class label. Machine learning techniques used for the substituting of missing values are given below:

- "Association Rules" based missing value filling algorithm is grounded on "Association Rules Mining" [20] for the substitution of missing values [9, 21, 22]. This technique is not suitable because it decreases the dataset diversity by filling single value for each missing item so each missing value can't be filled with the single rule. Moreover, the creation of a unique rule is not possible for each instance of missing value [9, 17, 18].

- "Clustering" based missing value filling algorithm is built on the "Clustering" approach [23] which fills the final missing values by substituting them with the replacement from the respective elements of the cluster [4, 24-27]. This is not a suitable approach for maintaining the dataset diversity during the missing values

filling by taking a single value from the whole cluster assuming that it will represent the cluster [17, 18].

- "SVM Regression" based missing value filling algorithm is built on the "Support Vector Machines (SVM)" [28] Regression model to fill the missing values from the already known values [14, 25, 29]. This technique has associated issues of incorrect missing values replacements because randomly selected single value from multiple values of a group or sub-group can't exactly substitute the missing value [17, 18].

- "Neural Networks" based missing value filling algorithm is built using "Artificial Neural Network" [30] algorithm to find the best suitable substitute for each missing value in the dataset [18, 31, 32]. It finds the exact replacement value by learning the data for each missing value one by one. There is a big issue associated with this technique that it requires a lot of computation power which is nearly impossible if the dataset has a large number of records having missing values [18].

- "K-NN" based missing value filling algorithm is built using "K-Nearest Neighbours" algorithm [33] and during this algorithm nearest neighbour of the missing value is searched and missing data is filled from the nearest neighbour [18, 34-38]. This technique is not suitable for large datasets because the blind comparison of each and every record with other records is not possible especially for the dataset having a large number of missing values [34].

- "Decision Tree" based missing value handling algorithm is built using the "Decision Tree" algorithm [39] but this technique is just limited to the handling of missing values. It is useful during the learning in a decision tree and not filling the missing values for the use with other classification algorithms. Moreover, this missing values handling mechanism is only checked for the decision tree not for the other classification algorithms [13, 18, 40, 41].

- "Naive Bayes" based missing value handling algorithm is built using the "Naive Bayes" classifier [42]. This technique calculates the replacement of missing value from the complete data using the probability of replacement value. It considers the

attributes independence for the selection of missing value replacement and was tested for the dataset having the independence of attributes [11, 15, 18, 43-45].

These techniques have associated issues of incorrect missing values replacements because randomly selected single value from multiple values of a group or sub-group can't exactly substitute the missing value [17, 18]. Moreover, the proposed algorithms were either tested for specific machine learning operators or developed for specific datasets like "Naive Bayes" based missing values filing algorithm was developed for "Naive Bayes" classifier and was tested for the dataset having the independence of attributes [11, 15, 18].

The correctness of machine learning based on missing values filling algorithms is directly dependent on the selection of appropriate replacement values. A single value for all the missing values disturbs the variability of a dataset and biases the accuracy of the classifier and can't indicate the variability of example and can affect the classification accuracy of dataset [2, 17-19].

Another limitation associated with the machine learning based on missing values filling algorithms is the complexity of calculations for a single value substitution. Filling the multiple values of multiple records require a lot of calculations and need a lot of resources for this calculation which is nearly impossible when data has missing values due to systemic error [18].

2.5. Review of Missing Values Filling Techniques

Techniques	Details	Problems
Ignore, Delete, Discard	The simpler method (for dealing with the missing values) is to delete, discard or ignore records containing the missing values while performing any data analysis on inconsistent data [13-15].	Records deletion can negatively influence the process of data mining because there is a chance to delete the records containing the most important data [16].

Techniques	Details	Problems
Single Imputations (SI)	This method specifies a particular value as a replacement of all missing values within a dataset. A single replacement value can be defined for all the missing values of all attributes (one value for all dataset) or multiple replacements can be defined for all the missing values of each attribute (one value for each attribute) [17].	This technique disturbs the diversity of a dataset and biases the accuracy of a classifier because single value replacement either on database level or on attribute level is neither an appropriate replacement nor a suitable representation of all the existing values of an attribute [6, 17, 18].
Mean (Type of SI)	This technique specifies an average or means value as a replacement of all missing values within a dataset. A single mean value is calculated for each attribute of the dataset and in this way multiple replacements are defined for the missing values of each attribute (single average value for each attribute) [5, 6].	This method considers the attributes independence which is not suitable for each and every dataset. It also disturbs the variability of a because a single value for all the missing values can't indicate the variability of the dataset and can affect the classification accuracy [12, 17, 19].
Hot Deck (Type of SI)	Hot deck fills the missing value by getting the relevant replacement from the same single dataset which means replacing and replaced records are found on the same dataset [6, 17].	This technique disturbs the diversity of a dataset and biases the accuracy of a classifier because randomly selected single replacement value from the same dataset is not an appropriate replacement of all the existing values of an attribute [2, 17].

Techniques	Details	Problems
Cold Deck (Type of SI)	Cold deck fills the missing value by getting the relevant replacement from the different dataset which means replacing and replaced records are found on the different dataset [18].	This technique disturbs the diversity of a dataset and biases the accuracy of a classifier because randomly selected single replacement value from the different dataset is not a suitable representation of all the existing values of an attribute [2, 17].
Most Likely (Type of SI)	This methodology considers each attribute (one by one) and calculates their missing values' replacements from the non-null values. Maximum occurred non-null attribute's value is selected as replacement of missing values for that attribute [5, 6, 19].	This method considers the independence of attributes which is not suitable for each and every dataset. It also disturbs the variability of a dataset and biases the accuracy of classifiers when data mining is performed. A single value either for the complete dataset or for a specific subset of data can affect the variety of dataset and can disturb the classification accuracy [12, 17, 19].
Not Applicable (Type of SI)	This is a special method to treat the missing and null values of the dataset by substituting them with a predefined tag similar to "Not Applicable", "N/A", "Undefined" or "Not Available" [4]. It is a suitable methodology when some dataset can't be filled for its missing values	This method is not suitable for each and every dataset and disturbs the variability of a dataset and biases the accuracy of classifier when data mining is performed. A single value for all the missing values can't indicate the variability of example

Techniques	Details	Problems
	since some values remain not applicable in some instances and applicable for some instances[6].	and can affect the classification accuracy of the dataset [17, 19].
Multiple Imputations (MI)	Total ***x*** number of database copies are created for ***x*** number of arithmetically calculated replacements for missing values. Each copy of dataset is passed through the numerical investigation, arithmetical examination and "Exception Maximization" calculates the value of probable replacement similar to the Bayesian method and final missing value replacement is selected [17, 18].	This process reflects the individuality of attributes which is not appropriate for all types of datasets. It also disturbs the variability of a dataset and biases the accuracy of classifiers when data analysis is performed. Specific values either from the complete dataset or from a specific subset of data can affect the variety of dataset and can disturb the classification accuracy [17, 19].
Machine Learning (ML)	Researchers presented methods to fill the missing values by creating the subsets of dataset like subgroups having a similar class label or clusters of similar records [13, 15]. This methodology considers a missing value record as test data and trains the model to find the missing value attribute as class label.	A limitation associated with the machine learning based on missing values filling algorithms is the complexity of calculations for a single value substitution. Filling the multiple values of multiple records require a lot of calculations and need a lot of resources for this calculation which is nearly impossible when data has missing values due to systemic error [18].

Techniques	Details	Problems
Association Rules (Type of ML)	"Association Rules" based missing value filling algorithm is grounded on "Association Rules Mining" [20] for the substitution of missing values [9, 21, 22].	Creation of unique rule is not possible for each instance of missing value [9, 17, 18]
Clustering (Type of ML)	"Clustering" based missing value filling algorithm is built on the "Clustering" approach [23] which fills the final missing values by substituting them with the replacement from the respective elements of the cluster [4, 24-27].	This is not a suitable approach for maintaining the dataset diversity during the missing values filling by taking a single value from the whole cluster assuming that it will represent the cluster [17, 18].
SVM Regression (Type of ML)	"SVM Regression" based missing value filling algorithm is built on the "Support Vector Machines (SVM)" [28] Regression model to fill the missing values from the already known values [14, 25, 29].	This technique has associated issues of incorrect missing values replacements because randomly selected single value from multiple values of a group or sub-group can't exactly substitute the missing value [17, 18].
Neural Networks (Type of ML)	"Neural Networks" based missing value filling algorithm is built using "Artificial Neural Network" [30] algorithm to find the best suitable substitute for each missing value in the dataset [18, 31, 32].	There is a big issue associated with this technique that it requires a lot of computation power which is nearly impossible if the dataset has a large number of records having missing values [18].

Techniques	Details	Problems
K-Nearest Neighbors (Type of ML)	"K-NN" based missing value filling algorithm is built using "K-Nearest Neighbors" algorithm [33] and during this algorithm nearest neighbour of the missing value is searched and missing data is filled from the nearest neighbour [18, 34-38].	This technique is not suitable for large datasets because the blind comparison of each and every record with other records is not possible especially for the dataset having a large number of missing values [34].
Decision Tree (Type of ML)	"Decision Tree" based missing value handling algorithm is built using the "Decision Tree" algorithm [39] but this technique is just limited to the handling of missing values.	It is useful during the learning in a decision tree and not filling the missing values for the use with other classification algorithms. Moreover, this missing values handling mechanism is only checked for the decision tree not for the other classification algorithms [13, 18, 40, 41].
Naive Bayes (Type of ML)	"Naive Bayes" based missing value handling algorithm is built using the "Naive Bayes" classifier [42]. This technique calculates the replacement of missing value from the complete data using the probability of replacement value.	It considers the attributes independence for the selection of missing value replacement and was tested for the dataset having the independence of attributes [11, 15, 18, 43-45]

2.6. Duplicate Records

The second part discovers the tools and algorithms associated with the detection and cleansing of duplicate records from the datasets. Duplicate data affect the quality of data and create problems for successful data mining process. The dataset inconsistencies lead to unreliable and biases data mining results [1] and these data irregularities particularly affect the process of supervised learning [4]. Therefore, duplicates records need to be cleaned prior to making decisions, predictive analysis and forecasting. Duplicate cleansing is a process of duplicate record detection and merging. Duplicate detection discovers groups of records which belong to the same real-world entity [7, 46]. Duplicate fusion is one of the duplicate cleansing process developed to clean the duplicate records from the datasets [46-48].

Machines (i.e. sensors, devices) and humans (i.e. filled forms, saved data) are the key sources of data created in this digital era. Digital systems administrators are authorizing devices to record each and every minute and relevant instance in their storage medium. A record in the database is created by combining the values of different attributes from similar or different sources. Sometimes, machines receive an identical or parallel event which is logged twice as duplicate data by single or multiple machines in the same recording medium.

2.7. Categories of Duplicate Data

Duplicate data issues are caused due to machine error like sensor error and these issues also exist in human-generated data mostly due to human error, typo errors or other input device related errors. This research divides the data duplication issues into two main categories based on the reasoning behind duplicate data as given below:

- Deliberate Duplicates
- Unintended Duplicates

1) Deliberate Duplicates

Sometimes, duplicate readings or records are generated intentionally just to create a more valid and trustworthy end result from the recordings of sensors or recording sources. Sensors or recording mediums are paced in parallel to create a more useful output from the recorded data. Deliberate duplication is carried out either to record a similar event or to record the different overlapping events [49-52].

Decisions on the given input within the critical systems can't be based on a single source of input so multiple parallel sensors or input sources are placed to record the same event from the environment which leads to deliberate duplication [49, 52].

In critical systems, a single sensor can't be dependable to create a complete picture before taking any decisions on the input, therefore, parallel sensors are placed to record the duplicate input from the same event happening in the environment. Robots take the decision to perform an action based on the several inputs received from the multiple sources about the happening of a single event [50, 51].

In some systems, a single sensor can't record each and every aspect of a single event, therefore, overlapping sensors are installed to create a complete picture before taking any decisions on the input. Overlapping recording sources such as sensors placed in the body of automatic car receive overlapping inputs to take the decisions related to taking a turn or applying breaks [49, 52, 53].

Online search results are created by finding and placing the identical results together just to give the range of available options to the user. This is also an example of deliberate duplication and it is useful to collect similar records which increase the chances to get valuable search results [54].

2) Unintended Duplicates

Sometimes, duplicate readings or records are generated unintentionally by the sensors or recording sources when the similar event is recorded twice by the similar or identical sensor. These duplicates are not necessarily the exactly identical of each other but represent the same entity or the event which is recorded twice by the system [7]. These duplicates are caused due to machine error like sensor error or other input device related errors.

These anomalies are divided into two categories i.e. systematic irregularities or the non-random inconsistencies and non-systematic irregularities or the random inconsistencies [1-3]. Systematic irregularities or non-random inconsistencies have some common error pattern and caused due to device error or device limitation to record specific types of incidents. Non-systematic irregularities or random inconsistencies are occurred at a certain time only due to some rare event.

Systematic and non-systematic problems cause the irregularities of unintended duplicate record because duplicate records are generated during rewriting attempts for the events failed to be recorded [55, 56].

2.8. Duplicate Records Detection Techniques

Duplicate detection discovers groups of similar records which belong to the same real-world entity [7, 46] from the given input data. Duplicates records detection techniques are built on different duplicate records detection algorithms. Some of the famous duplicate records detection algorithms studied in this research are given below:

- "Naive Duplicate Detection" [7, 57]
- "Duplicate Count SNM" [7, 58]
- "Sorted Neighborhood Method" [7, 59]
- "Lego" [7, 60, 61]
- "GSwoosh" [7, 62]
- "RSwoosh" [7, 63]
- "Naive Blocking" [7, 64]

1) Naive Duplicate Detection

Naïve Duplicate Detection is a duplicate detection algorithm which imposes the Naive method of examination each likely duplicate pair. This algorithm limits the duplicate pairs with a logical way because a record can't be a duplicate pair with itself and can be a duplicate pair with another record just once means no [n,m] exists, if [m,n] is previously generated [7, 57].

2) Duplicate Count SNM

Duplicate Count SNM is a duplicate detection algorithm which imposes the Sorted Neighborhood Method based on Adaptive Window Size which means duplicate records window could be adaptive in terms of numbers of records taking part in that window. Sorted Neighborhood Method based on Adaptive Window Size was originally presented by Oliver Wonneberg [7, 58].

3) Sorted Neighborhood Method

Duplicate Count SNM is a simple duplicate detection algorithm which imposes the Sorted Neighborhood Method for the duplicate records detection. It allows the system to sort the records according to similarity to avoid multiple executions on the input day to find the duplicate pairs from the sorted neighbours [7, 59].

4) Lego

Lego is a duplicate detection algorithm which imposes the approach of iterative blocking. Duplicates blocks are generated iteratively and duplicate records are spread to the different blocks to get the more duplicate pairs. Due to iterative nature, this process is repeated until duplicates are found in each previous iteration. This process is based on "Entity Resolution with Iterative Blocking" by "Steven Euijong Whang", "David Menestrina", "Georgia Koutrika", "Martin Theobald" and "Hector Garcia-Molina" [7, 60, 61].

5) GSwoosh

GSwoosh is a duplicate detection algorithm which imposes the approach of "Swoosh: a generic approach for entity resolution." Similarity Function is integrated with this approach to enable it to utilize the Cross Product Strategy [7, 62]. This algorithm is also used for duplicate fusion tasks.

6) RSwoosh

RSwoosh is a duplicate detection algorithm which imposes the approach of "Swoosh: a generic approach for entity resolution." Similarity Function is integrated with this approach to enable it to utilize the Cross Product Strategy Additionally, RSwoosh considers the attributes of "idempotence", "commutativity", "associativity" and "representativity" [7, 63].

7) Naive Blocking

Naïve Blocking Algorithm is a duplicate detection algorithm which imposes the Naive blocking method of examination each likely duplicate pair. This algorithm limits the

duplicate pairs with a logical way using Sorting Key for the Sorted Blocks and this sorting key is generated uniquely for each block [7, 64].

8) Similarity Comparison Algorithms

Different algorithms used for the calculation of similarities of the generated duplicate pairs. Similarities of all possible pairs are calculated with the algorithms given below:

- "Levenshtein Distance Function" [7, 65]
- "Euclidean Distance Function" [7, 66]
- "Jaro Winkler Function" [7, 67]
- "Jaccard Similarity Function" [7, 68]
- "Cosine Similarity Function" [7, 69]
- "Block Distance Function" [7, 70]
- "Dice Coefficient Function" [7, 71]

2.9. Duplicate Records Fixing Techniques

Duplicate records in a dataset lead to unreliable and biases data mining results [1] and particularly affect the process of supervised learning [4]. Therefore, duplicates records need to be cleaned prior to making decisions, predictive analysis and forecasting from any input data. Manually fixing the dataset inconsistencies is completely unmanageable due to the size of the dataset and because of the complications connected with inconsistencies. Data preprocessing techniques [8, 9] were introduced to fix the irregularities of the datasets by eliminating or replacing the duplicate records. Following are the two main approaches to fix the issues related to duplicate records in a dataset:

- Delete, Discard Duplicate Records
- Duplicate Records Fusion

1) Delete, Discard Duplicate Records

The simplest method for dealing the duplicate records of a dataset during which records containing the similar values are deleted or discarded from the dataset create the clean

dataset. Most commonly records having most erroneous values are discarded to leave and more stable dataset behind [50, 51]. This duplicate removal method is more suitable for the unintended duplicates generated due to some error.

This technique has different variations based on the selection mechanism to discard or delete an item:

- Keep the record having the latest or oldest data [72-74]:
    - In this technique, the record having the latest or oldest data from the duplicate records group is kept in the cleansed dataset and remaining duplicate records are removed from the dataset. These decisions are taken based on the date of record creation or modification [72-74].
- Keep the record having least null values and discard others [2, 14, 72, 75]:
    - In this technique, the record having not null values from the duplicate records group is kept in the cleansed dataset and remaining duplicate records are removed from the dataset [2, 14, 72, 75].
- Keep the record having less erroneous values and discard others [2, 14, 72]
    - In this technique, the record having less erroneous values from the duplicate records group is kept in the cleansed dataset and remaining duplicate records are removed from the dataset [2, 14, 72].

Deleting duplicates is preferred when the dataset contains exact duplicate records with greatest similarity value. This technique is not suitable for the dataset containing duplicate records with low similarity value because in such datasets a single record can't represent all discarded records and during the records discarding important information could be lost. A single value for all the missing values can't indicate the variability of the dataset and can affect the classification accuracy [12, 17, 19].

2) Duplicate Records Fusion

Duplicate fusion is one of the duplicate cleansing process developed to clean duplicate records from the datasets [46-48, 75]. During the data fusion, duplicate records are merged to create a new and more accurate and useful record as a result. Duplicate fusion technique

not only resolves the unintended duplicates issue but also helps to merge the deliberately created duplicates from the parallel sensors setup [49-52, 76, 77]. This technique is widely used for the fusion of multiple inputs received from the multiple sensors installed to record a similar event [50, 78].

Decisions on the given input within the critical systems can't be based on a single source of input so multiple parallel sensors or input sources are placed to record the same event from the environment which leads to deliberate duplication [49, 52]. Multiple inputs from the multi-sensor environment are merged to create a single and more useful record to take the decisions in the critical system [49, 51, 52, 77].

In some systems, a single sensor can't record each and every aspect of a single event, therefore, overlapping sensors are installed to create a complete picture before taking any decisions on the input. Overlapping recording sources such as sensors placed in the body of automatic car receive overlapping inputs to take the decisions related to taking a turn or applying breaks [49, 52, 53]. Partial inputs from the multi-sensor environment are merged to create a single and more useful record to take the decisions about the events and to predict the future events and consequences [49, 51, 52, 77].

3) Voting-Based Duplicate Fusion:

Majority of the researches focus on voting based duplicate fusion technique to merge the duplicate records in the dataset. Multiple records representing the same real-world entity are merged in duplicate fusion to create more accurate representation on record. Voting technique consider the is attributes values as a candidate for vote and records with similar to this are considered as a vote in its favour, finally merging is done by picking the values with maximum votes as a proof of value validity [50, 73, 78-82].

i. Accuracy-Based Voting

Accuracy based voting technique consider the accuracy of the source from where records were taken, means the previous accuracy of sensors is considered as a voting mechanism [80, 81]. The voting number is equal to the accuracy of each attribute and is named as naive accuracy similar to "Naive Bayes" where consideration of attributes independence is a basic principle [80, 81, 83, 84]. Other than accuracy precision, f-measure or recall could also be used for this voting

method [84]. This technique has the limitation of getting the accuracy before filling missing values and this technique becomes less useful in case of absence of accuracy value.

ii. Similarity-Based Voting

Cosine similarity among records is calculated to and each record vote for other records equal to its similarity and at the end voting decide the reliability of any record. All records within all duplicate groups are passed through this mechanism and final record is created from each duplicate group after voting [81, 83]. This technique has one limitation of neglecting a number of occurrences because more occurrences could be overlapped with a value. It neglects the data mining principle of "wisdom of the crowd" [85, 86] in terms of occurrences by replacing the value having more occurrences with possibly some erroneous value.

iii. Weightage-Based Majority Voting

Some researchers suggested adding a weight to vote which means multiple weights criteria decide the final items from voting. Weightage could be based on similarity or accuracy or the trustworthiness and at the end, voting decides the final output from the fusion operation [54, 83, 87]. It neglects the data mining principle of "wisdom of the crowd" [85, 86] in terms of occurrences by replacing the value having more occurrences with possibly some erroneous value.

2.10. Review of Duplicate Cleansing Techniques

Techniques	Details	Problems
Keep the record having the latest value **(Keep one, discard others)**	The record having the latest data from the duplicate records group is kept in the cleansed dataset and remaining duplicate records are removed from the dataset. These decisions are taken based on the date of record creation or modification [72-74]. Deleting	This technique is not suitable for the dataset containing duplicate records with low similarity value because in such datasets a single record can't represent all discarded records and during the records

Techniques	Details	Problems
	duplicates is preferred when the dataset contains exact duplicate records with greatest similarity value.	discarding important information could be lost [12, 17, 19].
Keep the record having the oldest value **(Keep one, discard others)**	In this technique, the record having the oldest data from the duplicate records group is kept in the cleansed dataset and remaining duplicate records are removed from the dataset. These decisions are taken based on the date of record creation or modification [72-74]. Deleting duplicates is preferred when the dataset contains exact duplicate records with greatest similarity value.	This technique is not suitable for the dataset containing duplicate records with low similarity value because in such datasets a single record can't represent all discarded records and during the records discarding important information could be lost [12, 17, 19].
Keep the record having least null values and discard others **(Keep one, discard others)**	The record having not null values from the duplicate records group is kept in the cleansed dataset and remaining duplicate records are removed from the dataset [2, 14, 72, 75]	A single value from the containing duplicate records with low similarity value for all the missing values can't indicate the variability of the dataset and can affect the classification accuracy [12, 17, 19].
Keep the record having less erroneous values and discard others	In this technique, the record having less erroneous values from the duplicate records group is kept in the cleansed dataset and remaining duplicate records are removed from the dataset [2, 14, 72]	A single value from the containing duplicate records with low similarity value for all the missing values can't indicate the variability of the dataset and can affect the

Techniques	Details	Problems
(Keep one, discard others)		classification accuracy [12, 17, 19].
Accuracy-Based Voting (Fusion)	Accuracy based voting technique consider the accuracy of the source from where records were taken, means the previous accuracy of sensors is considered as a voting mechanism [80, 81]. The voting number is equal to the accuracy of each attribute and is named as naive accuracy similar to "Naive Bayes" where consideration of attributes independence is a basic principle [80, 81, 83, 84]. Other than accuracy precision, f-measure or recall could also be used for this voting method [84].	This technique has the limitation of getting the accuracy before filling missing values and this technique becomes less useful in case of absence of accuracy value.
Similarity-Based Voting (Fusion)	Cosine similarity among records is calculated to and each record vote for other records equal to its similarity and at the end voting decide the reliability of any record. All records within all duplicate groups are passed through this mechanism and final record is created from each duplicate group after voting [81, 83]	It neglects the data mining principle of "wisdom of the crowd" [85, 86] in terms of occurrences by replacing the value having more occurrences with possibly some erroneous value.
Weightage-Based	Some researchers suggested adding a weight to vote which means multiple	It neglects the data mining principle of "wisdom of the

Techniques	Details	Problems
Majority Voting (Fusion)	weights criteria decide the final items from voting. Weightage could be based on similarity or accuracy or the trustworthiness and at the end, voting decides the final output from the fusion operation [54, 83, 87]	crowd" [85, 86] in terms of occurrences by replacing the value having more occurrences with possibly some erroneous value.

Chapter 3

PROPOSED APPROACH

This chapter discusses the solution developed as a result of research on data quality issues. The developed system named **Data Cleansing System (DCS)** attempts to solve the dataset quality problems for missing and duplicate values are presented in this chapter. The developed system is based on duplicate detection mechanism during which record having some similarities (duplication) with each other are extracted in the form of pairs named as duplicate pairs.

3.1. Development Process

An iterative agile methodology named Solo Iterative Process (SIP) [88, 89] is followed to develop the Data Cleansing System (DCS). During the development process, the complete system is divided into four main subsystems based on the independence of functionalities. Each subsystem is further divided into smaller modules and tasks based on the sub-functionalities. Division of functionalities and sub-functionalities was performed to make the work more manageable and agile.

Data Cleansing System (DCS) is developed in two iterations to reduce the gaps in the software development process [90, 91]. During the first iteration, basic functionalities are implemented to create a preliminary end to end system. During the second iteration, advanced functionalities are added to the system by extending the existing developed functionalities. Process flow details of each submodule with respect to iteration are discussed in the next sections.

3.2. Data Cleansing System (DCS)

A data quality improvement system, named as **Data Cleansing System (DCS)**, is developed during this research to fill the missing or null values of a dataset and to merge the duplicate records of the dataset. The developed system is based on duplicate detection mechanism during which record having some similarities (duplication) with each other are extracted in the form of pairs named as duplicate pairs. The extracted duplicate pairs are utilized to take the duplicate merging decisions by the system. Moreover, the developed system also utilizes these duplicates pairs to fill the missing values within the pairs.

Java programing language is used to build the Data Cleansing System (DCS) by extending the Duplicate Detection Toolkit (DuDe) [7]. The developed system takes the low-quality data as input, processes it, fills its missing values, removes its duplicates and generates the cleaned output data with improved quality.

3.3. Data Cleansing System (DCS) Architecture

The high-level architecture of Data Cleansing System is given below in Fig. 2. The developed system named Data Cleansing System (DCS) attempts to solve the dataset quality problems for missing and duplicate values are presented in this chapter.

The developed system detects the duplicates of records from the dataset during which record having some similarities (duplication) with each other are extracted in the form of pairs named as duplicate pairs. A threshold is defined within the system by the system administrator to limit the similarity comparison which decreases the duration of execution time.

- The raw dataset file is given as input to the system which is sent to the subsystem of **Pre-Processing** to extract and select the attributes from the dataset.
- The data is extracted for the selected attributes which are then passed to the **Duplicate Detection** subsystem to extract the duplicate pairs from the dataset.
- The extracted duplicate pairs are then passed to the **Duplicate Fusion** subsystem to fuse the duplicates into a single record.
- The fused duplicates are then passed to the **Post-Operation** subsystem to create the cleaned copy of input dataset by adding attribute names and non-duplicate data in it.

3.4. Flow Chart of Data Cleansing System (DCS)

Flowchart representing the sequence of the processes carried out by Data Cleansing System is given in Fig. 3. After processing of data in one subsystem, it moves to the next subsystem for further processing and this process ends with writing the cleaned data in the output file. The output of one subsystem is provided as the input to the next subsystem. Upon some unsuccessful operation or exception, subsystem exists to the start.

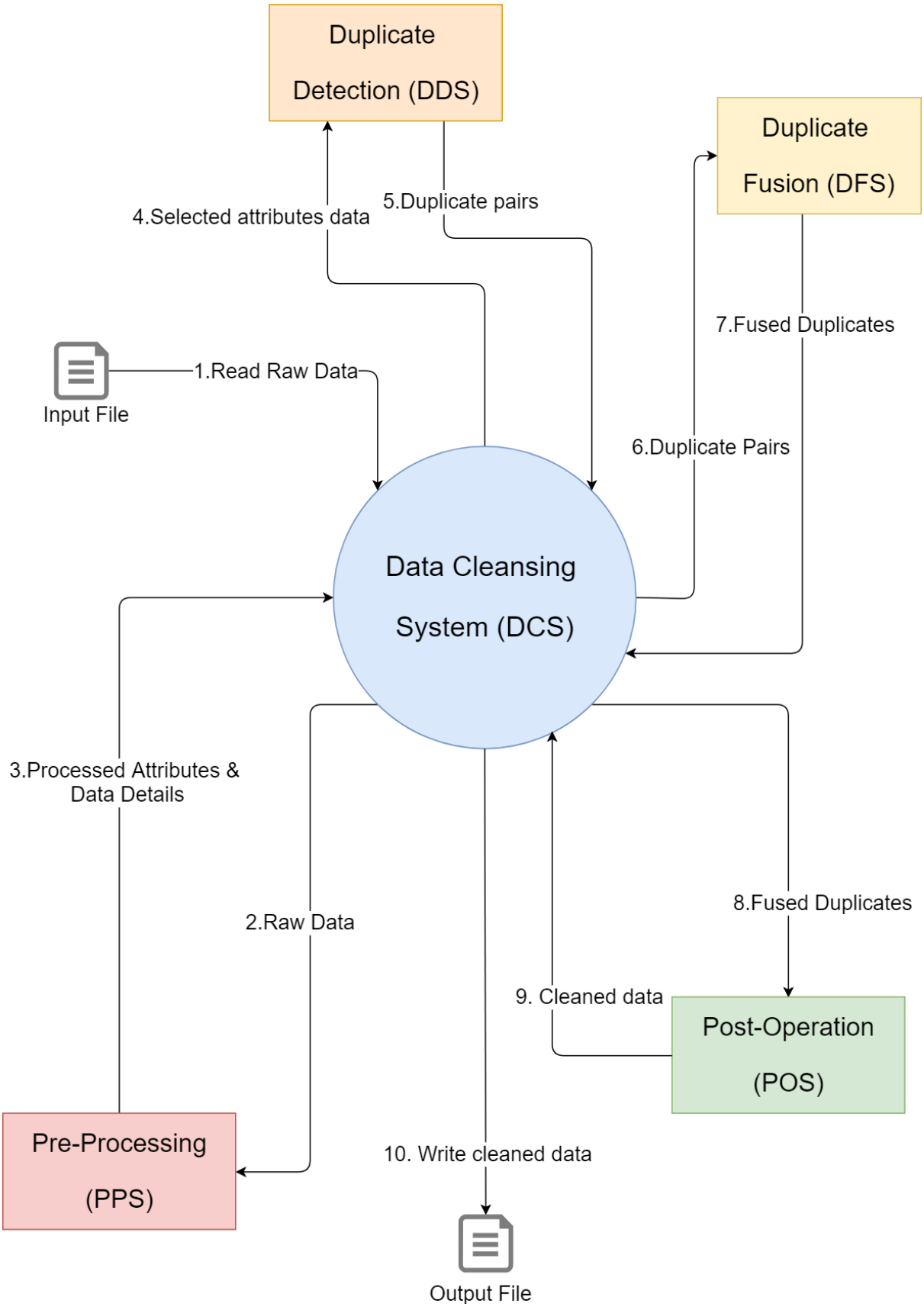

Fig. 2. High-Level Architecture of Data Cleansing System

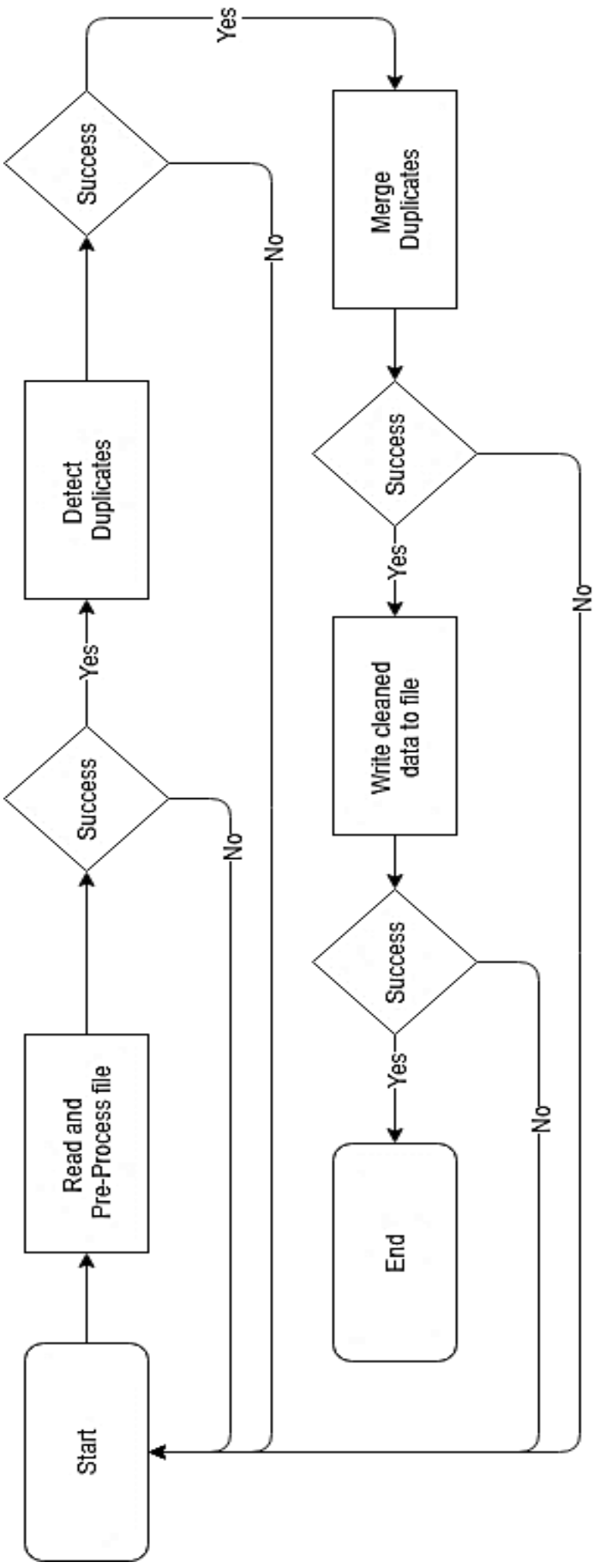

Fig. 3. Flowchart of Data Cleansing System

3.5. Sub-Systems of Data Cleansing System (DCS)

The higher level architecture of Data Cleansing System (DCS) is divided into four main subsystems based on the division of functionalities as shown in Fig. 4.

Four subsystems of DCS are:

- Pre-Processing Subsystem (PPS)
- Duplicate Detection Subsystem (DDS)
- Duplicate Fusion Subsystem (DFS)
- Post-Operation Subsystem (POS)

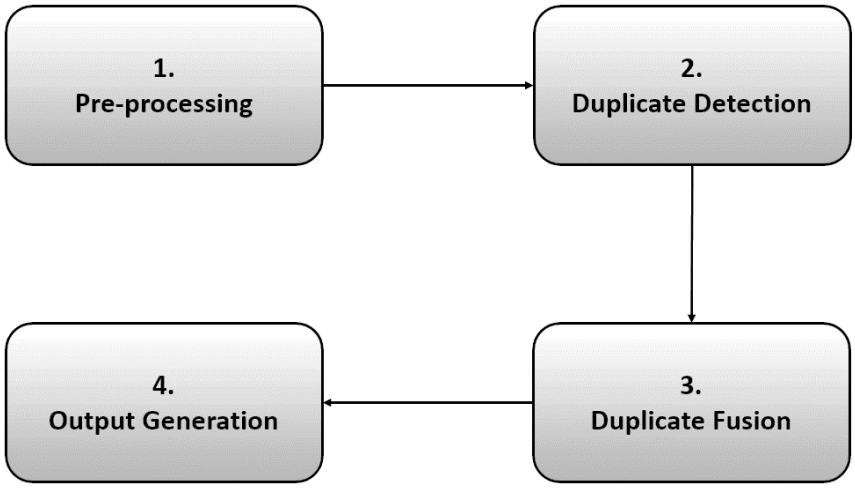

Fig. 4. Modules of Data Cleansing System

3.6. Pre-Processing Subsystem (PPS)

Pre-Processing Subsystem deals with the preliminary activities executed on the input file before performing any data cleansing task on it. PPS is further divided into four main modules as shown in Fig. 5. A user interface is created with an option to select and load the CSV file in the subsystem. After successful loading, the subsystem reads the input dataset from CSV file, extracts the list of attributes from it. The subsystem then asks the user to select the required attributes from the list of attributes and the selected attributes are used to extract the required data from the dataset which is returned to the main system for further processing.

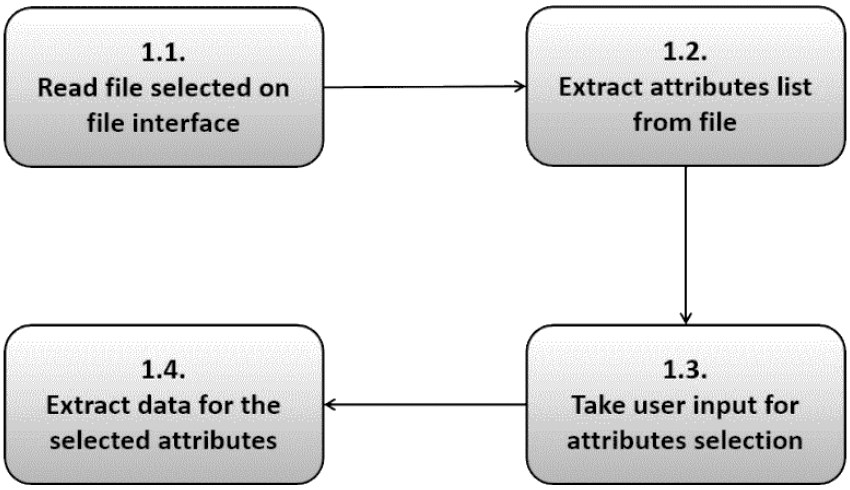

Fig. 5. Input File Upload and File Pre-Processing

3.7. PPS Flowchart

The sequence of steps performed by the developed Pre-Processing Subsystem is shown in Fig. 6. PPS is developed completely in a single iteration because it is just related to file loading and preprocessing functionality.

Details of the steps performed by the developed Pre-Processing Subsystem are given below:

- The developed PPS asks the user to select and upload the raw dataset CSV file into the subsystem.
- The user selects and uploads the file.
- PPS reads the file.
- PPS extracts the attribute list from a file.
- PPS reads data for all the extracted attributes of the file.
- PPS returns the extracted data and the attributes list to the Data Cleansing System for further processing and exists.

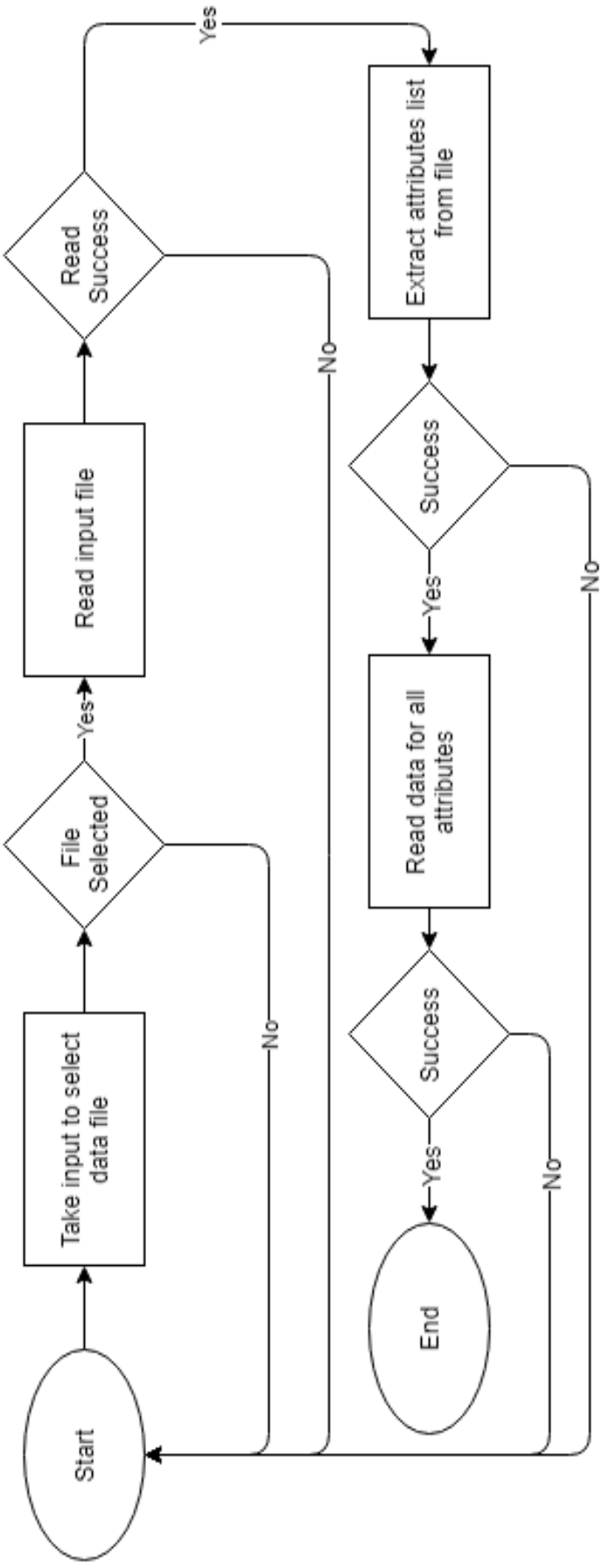

Fig. 6. Input File Upload and File Pre-Processing Flowchart

3.8. Duplicate Detection Subsystem (DDS)

After the successful preprocessing, extracted data and the attributes list is passed to the duplicate detection module to find duplicates from it as shown in Fig. 7.

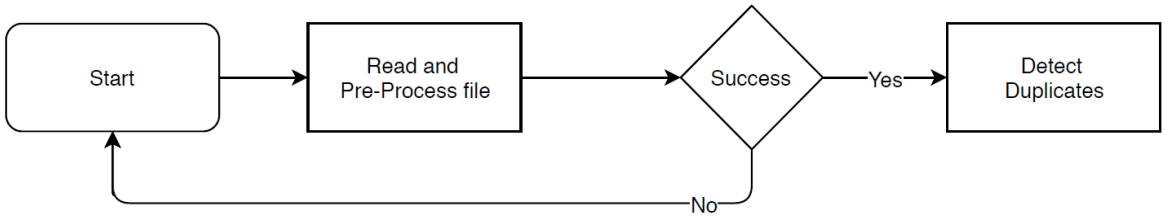

Fig. 7**.** Link of File Pre-Processing and Duplicate Detection

Duplicate Detection Subsystem (DDS) is further divided into four main modules as shown in Fig. 8. In the first module, the user is asked to set the similarity threshold after successful data extraction then records pairs are generated and their similarity is calculated. If pairs similarity exceeds the similarity threshold, pairs are saved for the later processing.

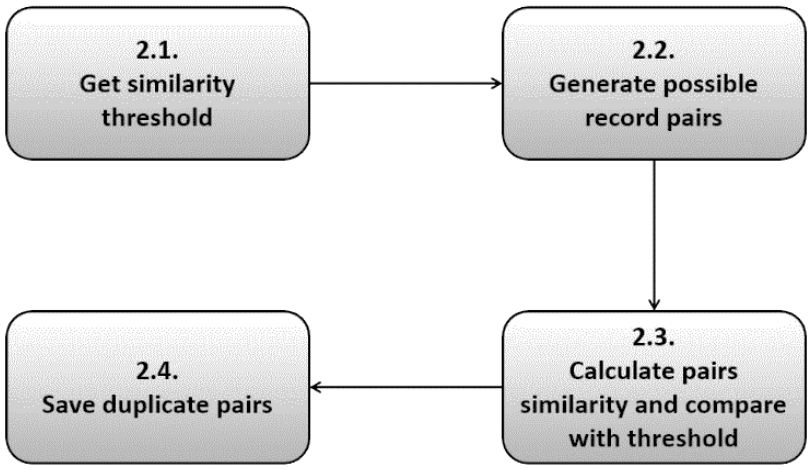

Fig. 8. Duplicate Detection with User Input and Rules

3.9. DDS Flowchart (First Iteration)

Fig. 9 shows the flow of a developed Duplicate Detection Subsystem during the first iteration which starts by picking the algorithm for duplicate detection and generates possible duplicate pairs after comparing records of the given dataset.

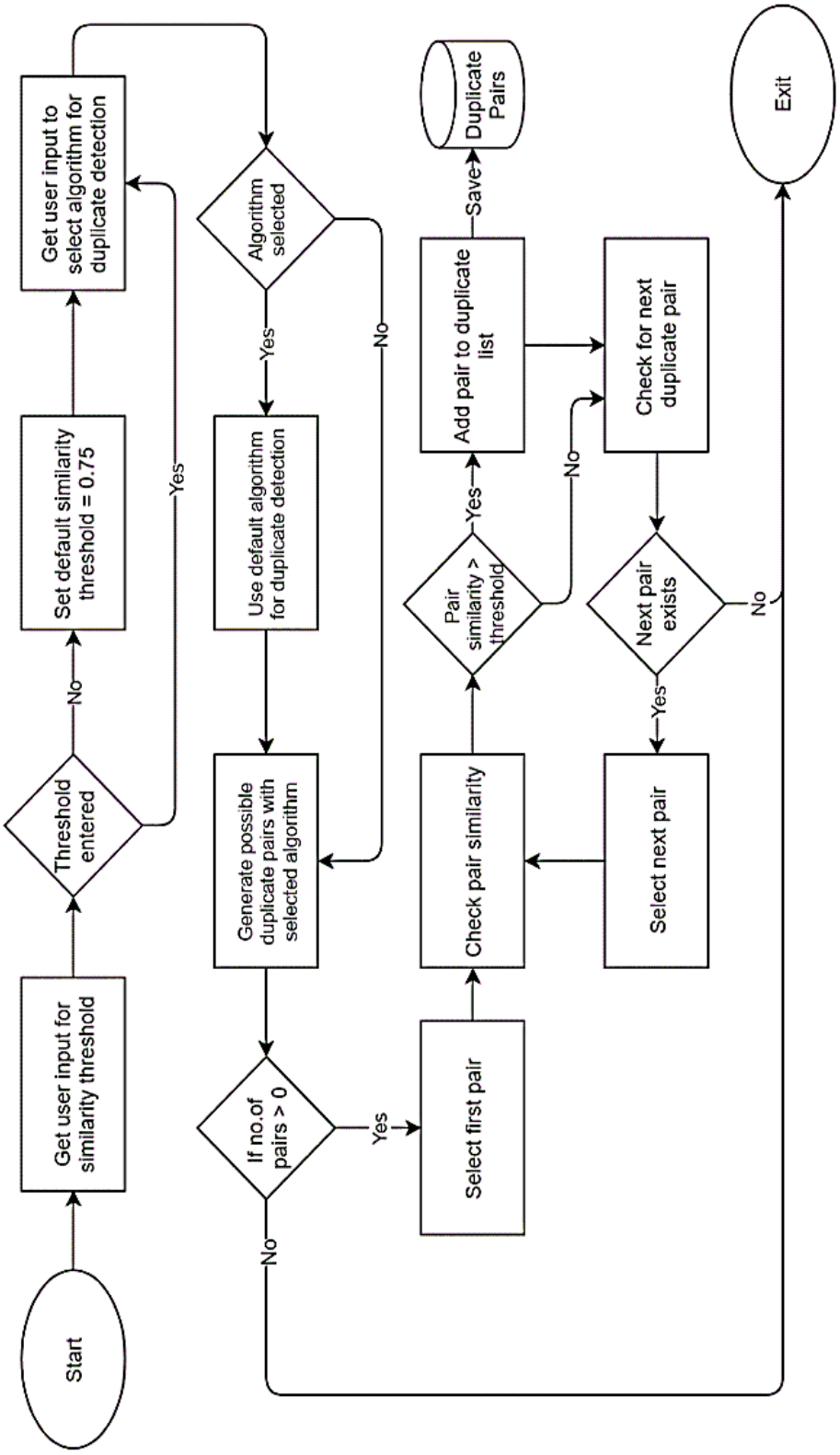

Fig. 9. Duplicate Detection Subsystem (First Iteration) Flowchart

These duplicate pairs are selected one by one and their similarity is calculated and compared with the similarity threshold defined in the system. During the comparison, duplicate pairs exceeding the similarity threshold are declared as confirmed duplicate pairs and are stored for further processing.

3.10. DDS Flowchart (Second Iteration)

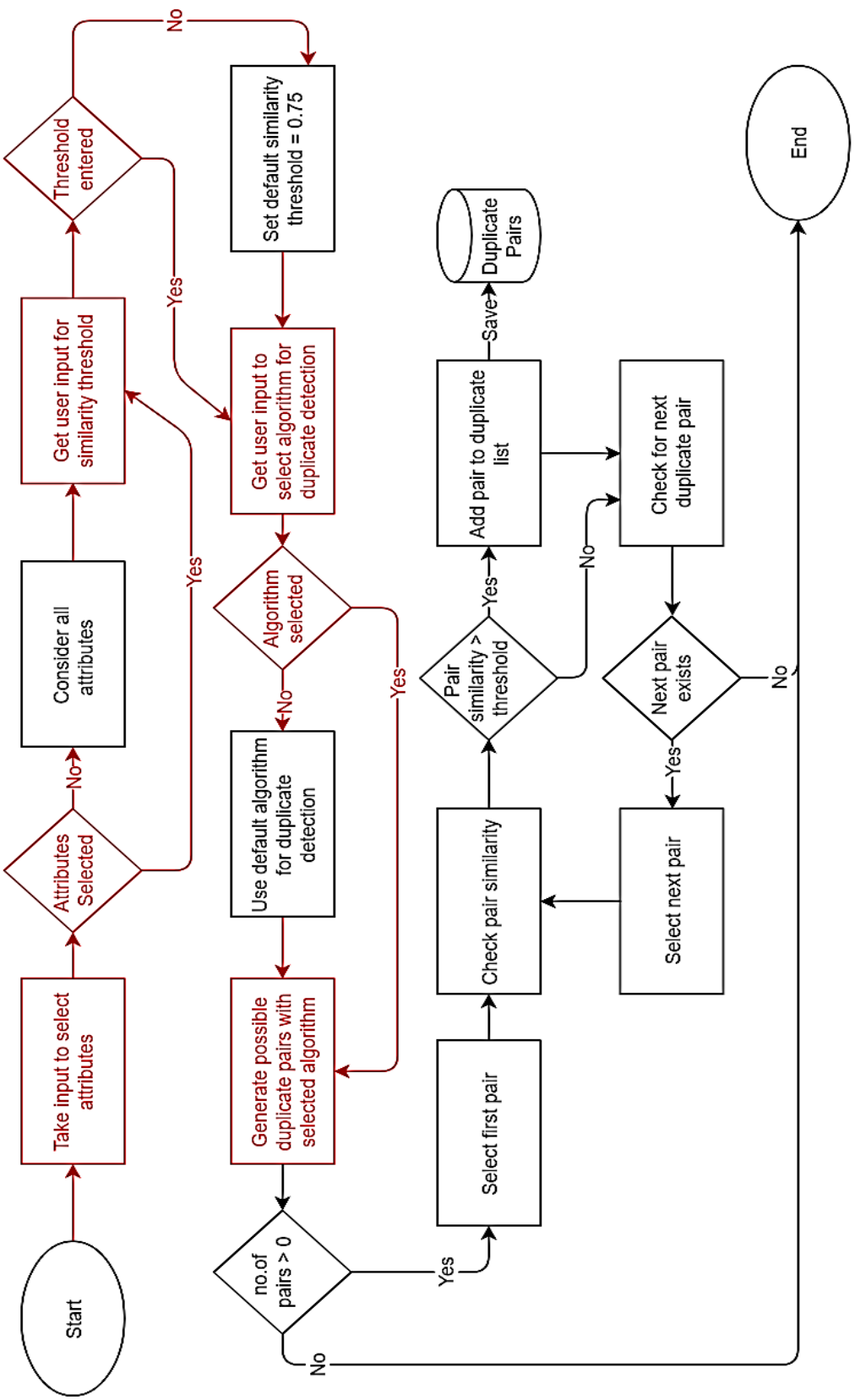

Fig. 10. Duplicate Detection Subsystem (Second Iteration) Flowchart

During the second iteration, the Duplicate Detection Subsystem is extended to get the user input for decisions related to duplicate detection.

Following options are added to the subsystem:

- Option to get similarity threshold for duplicate records comparison
- Option to get user input to select the attributes for duplicate detection
- Option to get user input to select the algorithm from the list of available algorithms specific to generate duplicate pairs
- Option to select the algorithm for calculating the similarity value of the generated duplicate pair

Duplicate Detection Subsystem is extended in such a way that lack of user input for duplicate detection decisions is not affecting the flow and subsystem is working by taking default actions to find duplicates as shown in the Fig. 10.

3.11. DDS Algorithms

Different algorithms for duplicate detection are studied from literature and the most discussed algorithms among researchers include Naive Duplicate Detection, Sorted Neighborhood Method, Naive Blocking Algorithm, Duplicate Count SNM (DCSNM) and DCS++. Duplicate Detection toolkit (DuDe) was developed as an open source platform which provided the Java source code implementation of the duplicate detection algorithms.

Java code of one of the algorithms named Naive Duplicate Detection [57] is integrated by taking it from the DuDe toolkit [7] because DuDe is open source toolkit. Java code given by the DuDe is also widely studied before integration to check for any anomalies and then it is extended further for the integration in the subsystem under development.

Correctness, accuracy and algorithmic efficiency of the code given by DuDe is widely studied by international researchers and they found it a trustable toolkit of Java source code for Duplicate Detection algorithms.

The Naive Duplicate Detection algorithm [57] is implemented initially and its results as detected duplicate pairs are validated with the DuDe and with the available literature. A variety

of duplicate detection algorithms express that a single duplicate detection algorithm is not suitable for all the scenarios and datasets which is also validated by many of the researchers in data cleansing community.

Moreover, the user should be able to select the desirable algorithm on runtime as per the need and based on the type of dataset given as input. Therefore, Architecture of Duplicate Detection is extended to support this need during the second iteration which still supports the default implementation.

There are two types of algorithms integrated into Duplication Detection Subsystem for the two different types of tasks performed by these algorithms:

1. Duplicate Pairs Generation
2. Similarities Calculation

1) Duplicate Pairs Generation

This category includes the algorithms used for the generation of duplicate pairs by comparing each record as a whole with the other records in the dataset. All possible pairs of each record are created with the algorithms given below:

- “Naive Duplicate Detection” [7, 57]
- “Duplicate Count SNM” [7, 58]
- “Sorted Neighborhood Method” [7, 59]
- “Lego” [7, 60, 61]
- “GSwoosh” [7, 62]
- “RSwoosh” [7, 63]
- “Naive Blocking” [7, 64]

The user is able to select any of the above duplicate detection algorithms at runtime to generate the duplicate pairs from the input data.

2) Similarities Calculation

This category includes the algorithms used for the calculation of similarities of the generated duplicate pairs. Similarities of all possible pairs are calculated with the algorithms given below:

- "Levenshtein Distance Function" [7, 65]
- "Euclidean Distance Function" [7, 66]
- "Jaro Winkler Function" [7, 67]
- "Jaccard Similarity Function" [7, 68]
- "Cosine Similarity Function" [7, 69]
- "Block Distance Function" [7, 70]
- "Dice Coefficient Function" [7, 71]

The user is able to select any of the above algorithms at runtime for the calculation of similarities of the generated duplicate pairs.

Architecture and flow of Duplicate Detection Subsystem are made generic to support any future algorithm added to the subsystem either related to pair generation or related to pairs similarity calculation.

3.12. Duplicate Fusion Subsystem (DFS)

In Duplicate Fusion Subsystem (DFS), extracted duplicate pairs are further analyzed to merge them into a single record. Duplicate fusion is one of the duplicate cleansing process developed to clean duplicate records from the datasets [46-48, 75]. During the data fusion, duplicate records are merged to create a new and more accurate and useful record as a result.

Duplicate fusion technique not only resolves the unintended duplicates issue but also helps to merge the deliberately created duplicates from the parallel sensors setup [49-52, 76, 77]. This technique is widely used for the fusion of multiple inputs received from the multiple sensors installed to record a similar event [50, 78].

The high-level architecture of the Duplicate Fusion is additionally divided into four modules as shown in Fig. 11.

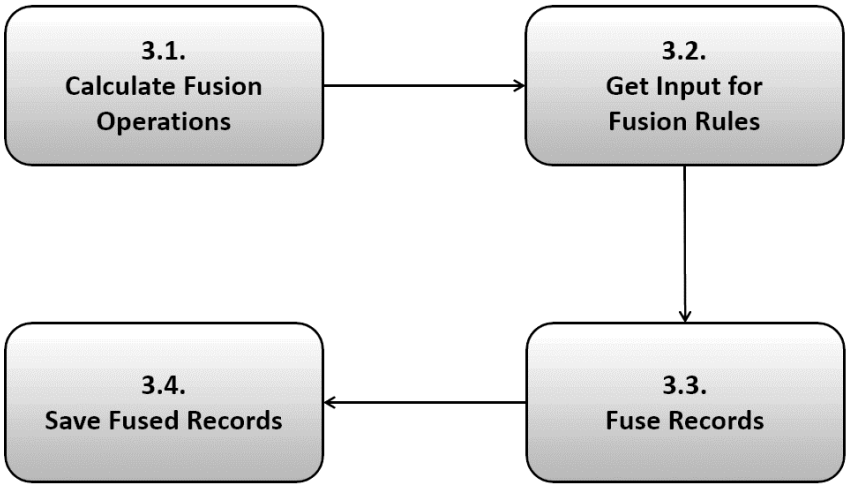

Fig. 11. Duplicate Fusion with User Input and Rules

Once fusion operations are calculated, subsystem prompts to get the user input related to merging procedures those are defined in the subsystem. When a user input is received for the fusion procedures, duplicate fusion is performed by merging the duplicate records and the dataset is updated. If no input is received default fusion operations are performed from the subsystem configuration.

3.13. DFS Flowchart (First Iteration)

Fig. 12 depicts the process of Duplicate Fusion Subsystem which is developed during the first iteration of DFS. Subsystem takes duplicate pairs as input, reads and selects each duplicate pair one by one and extracts the first record of duplicate pair. DFS then selects the all attributes one by one from the selected record and analyzes their value.

After the analysis, attribute's value is, its similarity value and occurrence are initialized and stored for that record if not already stored. Otherwise, the similarity value of the pair is added to the stored similarity of the record and the occurrence is incremented for that record if the attribute's value is already stored for that record. This process is repeated until all the duplicate pairs are processed and their respective replacements are calculated.

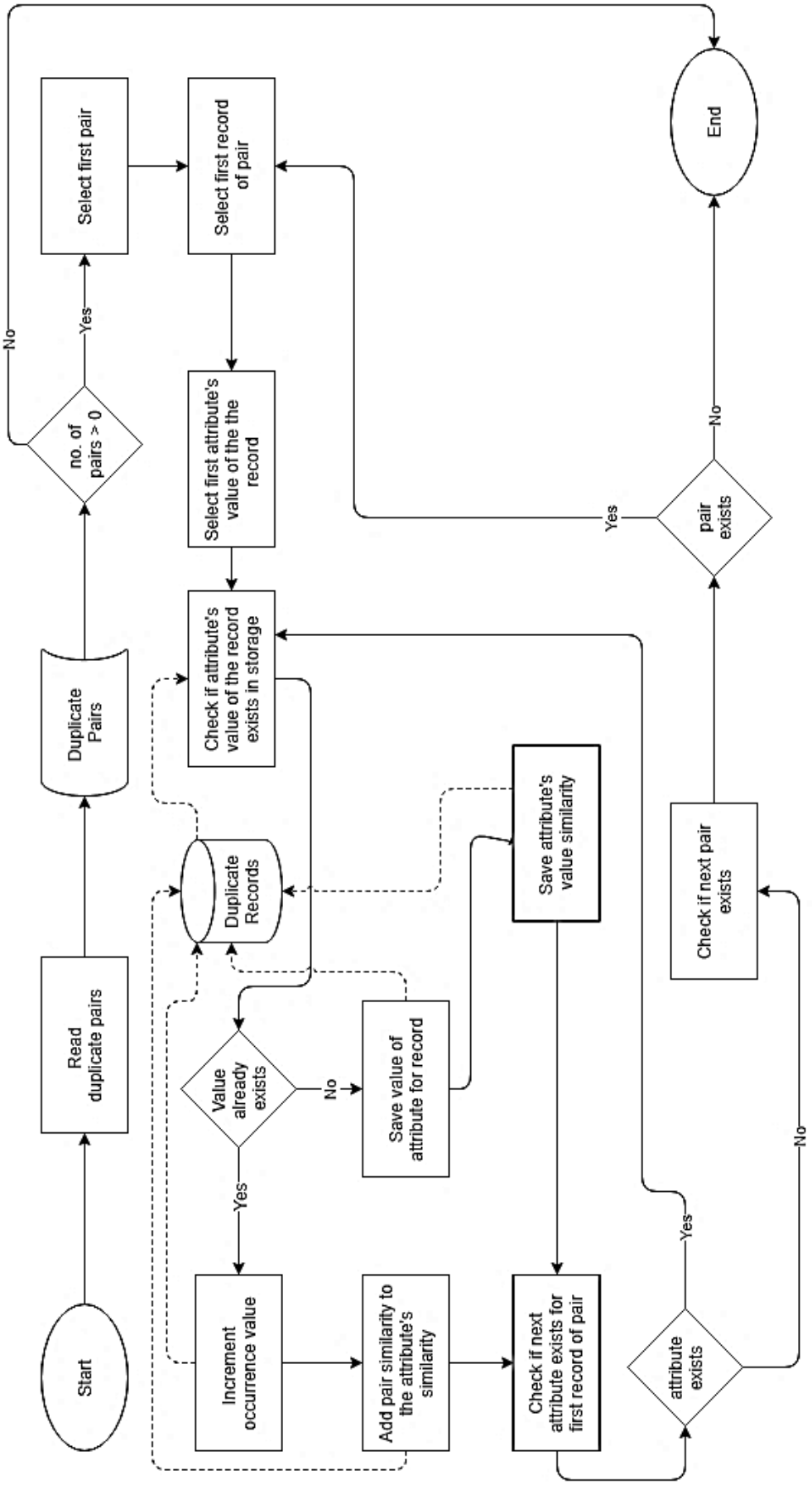

Fig. 12. Duplicate Fusion Subsystem (First Iteration) Flowchart

3.14. DFS Flowchart (Second Iteration)

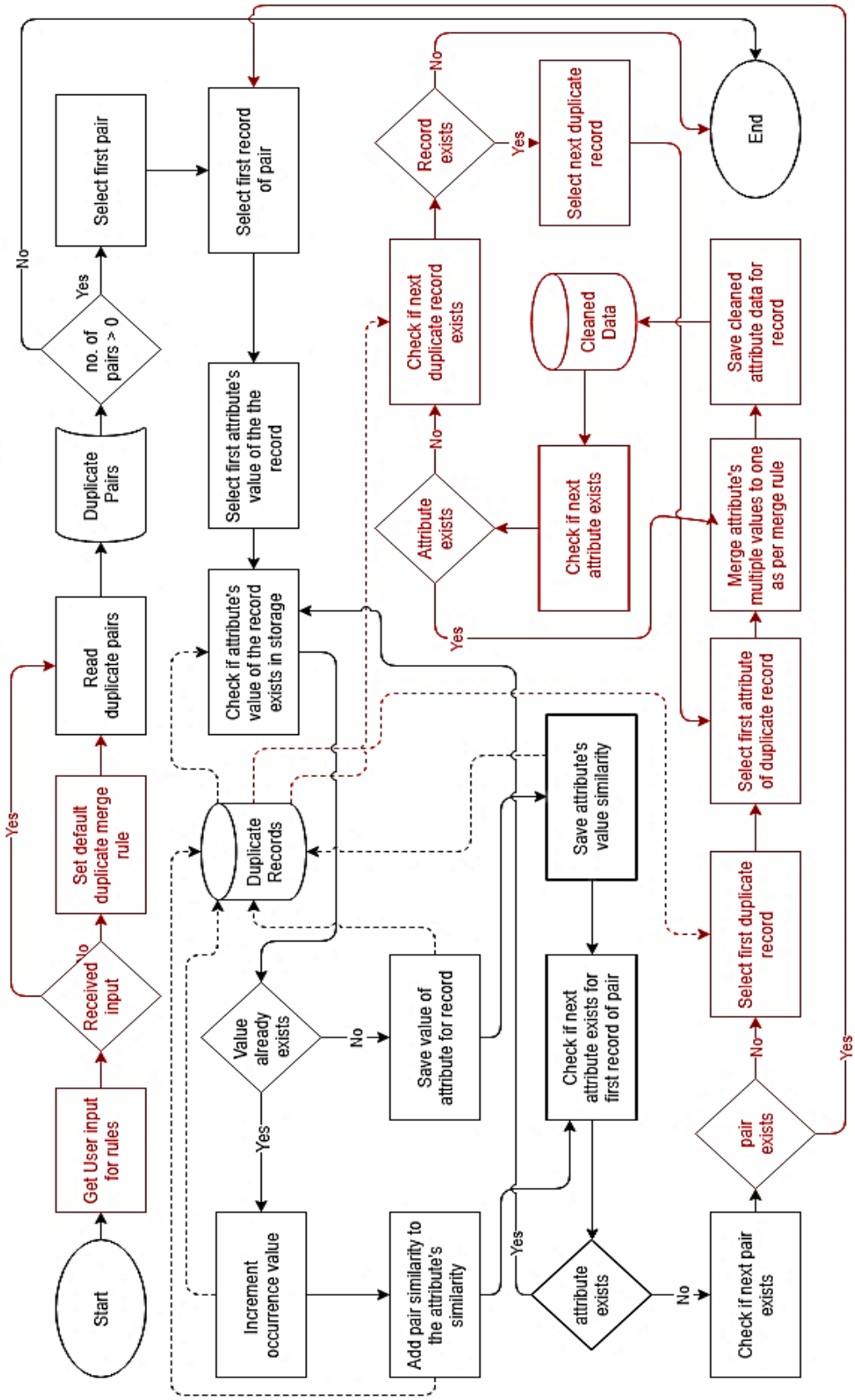

Fig. 13. Duplicate Fusion Subsystem (Second Iteration) Flowchart

Duplicate Fusion Subsystem (DFS) is further extended during the second iteration to get the user input related to the newly developed fusion options as shown in Fig. 13. The newly developed fusion options are based on the fusion operations calculated during the first iteration. After the selection of fusion options, all records are selected one by one and data cleansing is performed according to the values saved for each attribute of the selected record.

Duplicate Fusion Subsystem (DFS) performs one of the below three operations on the input duplicate pairs to perform the data cleansing as per the given user input. Runtime selection of one of these options is given to the user and the default option is executed on the system if nothing received from the user.

Duplicate Fusion Subsystem (DFS) supports the following three types of data fusion options:

1. Merge & Fill
2. Maximum Similarity Sum
3. Maximum Occurrences

3.15. DFS Fusion Options

The developed Duplicate Fusion Subsystem (DFS) provides three different types of duplicate fusion operations based on the calculations of fusion operations. These operations are carried out on the duplicate records those were detected during the first DFS iteration. Duplicate fusion operations implemented in the DFS are explained below with complete details.

1) Merge & Fill

This option of Duplicate Fusion Subsystem (DFS) supports the merging of highest similar duplicate record pairs just to fill the missing values of the records' attributes. During the merging and filling process, missing values of the records are filled and nothing is deleted from the dataset.

It is a useful technique for data preprocessing to fix the inconsistencies in the dataset related to missing values. It also improves the quality of the data after filling all possible missing values of a dataset.

2) Maximum Similarity Sum

This option of Duplicate Fusion Subsystem (DFS) supports the merging of duplicate record pairs by taking the value from the attributes having the highest similarity. It calculates the sum of similarity of all the values of an attribute from all relevant duplicate pairs. Records similar to each other participate in this process (duplicate pairs group) and create a new record by getting the values of highest accumulative similarity from each duplicate pair group.

3) Maximum Occurrences

This option of Duplicate Fusion Subsystem (DFS) supports the merging of duplicate record pairs by taking the value from the attributes having the highest number of occurrences. It calculates occurrences of all values of an attribute from all relevant duplicate pairs. Records similar to each other participate in this process (duplicate pairs group) and creates a new record by getting the values of highest occurrence from each duplicate pair group.

3.16. DFS Algorithms

Following three duplicate fusion algorithms are developed for the merging of duplicates and filling of missing values. These three options are integrated into the Duplicate Fusion Subsystem (DFS).

- SimFiller
    - Similarity-based missing values filling algorithm is developed to fill the missing values of any dataset.
- DuDeFiller
    - Duplicate detection based missing values filling algorithm is also developed to fill the missing values of any dataset.
- DuDeFuse
    - Duplicate detection and fusion algorithm are developed to fuse the duplicate values of any dataset.

3.17. SimFiller

SimFiller is similarity-based missing values filling algorithm which runs independently on its own to fill the missing values of a dataset because it is developed during the first iteration.

This algorithm directly accepts the input dataset after Pre-Processing Subsystem (PPS), produces the pairs of similar records, filters them based on similarity threshold, check each pair to verify that at least one record's attribute contains a null value then fills the missing values within each pair by taking the relevant not null value.

SimFiller technique is useful for data preprocessing to fix the inconsistencies in the dataset related to missing values. Quality of the dataset is increased after passing it from this data cleansing option.

Data passes through flowing steps during the cleansing with the SimFiller algorithm:

- Step1: Create pairs of similar data
- Step2: Select each pair and allocate it calculated a similarity value
- Step3: Save each pair having at least one non-null attribute's value
- Step4: Select a pair of highest similarity for each attribute's null value after matching pairs with similarity threshold and with each other.
- Step5: Replace all null values of attributes within a selected pair

3.18. Execution of SimFiller

SimFiller supports the merging of highest similar record pairs just to fill the missing values of the records' attributes. During the merging missing values of the records are filled and nothing is deleted from the dataset. This technique is useful for data preprocessing to fix the inconsistencies in the dataset related to missing values. Quality of the dataset is increased after passing it from this data cleansing option.

This algorithm takes the input dataset and passes it through the five steps process and generates the cleaned output dataset. Sample dataset containing missing values is considered to fill the missing values using this algorithm as shown in Table 1.

Similarity threshold is fixed at 0.75 for this sample execution and demonstrates that all the records having a similarity greater than 0.75 would be considered for filling the missing values of the sample dataset.

Considered sample dataset contains ten records (tuples) represented as $T_1 \ldots T_{10}$ and five attributes represented as $A_1 \ldots A_5$. After the execution of the first step of the algorithm on the sample dataset, similar pairs are generated and represented as $S_m = (T_n, T_o)$. Where **m** represents the number allocated to each record pair and **n** and **o** represent the record number i.e. $T_1 \ldots T_{10}$

	A_1	**A_2**	**A_3**	**A_4**	**A_5**
T_1	T_1A_1	T_1A_2	T_1A_3	T_1A_4	T_1A_5
T_2	T_2A_1	*	T_2A_3	T_2A_4	T_2A_5
T_3	T_3A_1	T_3A_2	T_3A_3	T_3A_4	T_3A_5
T_4	T_4A_1	T_4A_2	T_4A_3	*	T_4A_5
T_5	T_5A_1	T_5A_2	T_5A_3	T_5A_4	T_5A_5
T_6	T_6A_1	*	*	T_6A_4	T_6A_5
T_7	T_7A_1	T_7A_2	T_7A_3	T_7A_4	T_7A_5
T_8	T_8A_1	T_8A_2	*	T_8A_4	T_8A_5
T_9	T_9A_1	T_9A_2	T_9A_3	T_9A_4	T_9A_5
T_{10}	$T_{10}A_1$	$T_{10}A_2$	$T_{10}A_3$	$T_{10}A_4$	$T_{10}A_5$

Table 1. Sample Dataset (SimFiller)

1) Consider 1st missing value i.e. T_2A_2 to fill it with SimFiller

- **Step1:** Create pairs of similar data
  - Four pairs are created for T_2:
    - $S_1(T_2, T_4)$
    - $S_2(T_2, T_6)$
    - $S_3(T_2, T_9)$
    - $S_4(T_2, T_{10})$
- **Step2:** Select each pair and allocate it calculated a similarity value
  - $S_1(T_2, T_4) = 0.82$
  - $S_2(T_2, T_6) = 0.84$

- $S_3(T_2, T_9) = 0.80$
- $S_4(T_2, T_{10}) = 0.75$

- **Step3:** Save each pair having at least one non-null attribute's value
  - $S_2(T_2, T_6)$ omitted because T_6A_2 is null.
  - Remaining pairs:
    - $S_1(T_2, T_4) = 0.82$
    - $S_3(T_2, T_9) = 0.80$
    - $S_4(T_2, T_{10}) = 0.75$
- **Step4:** Select a pair of highest similarity for each attribute's null value after matching pairs with similarity threshold and with each other
  - $S_3(T_2, T_9)$ omitted because the similarity of $S_1(T_2, T_4) > S_3(T_2, T_9)$
  - $S_4(T_2, T_{10})$ is also omitted because the similarity of $S_1(T_2, T_4) > S_4(T_2, T_{10})$
  - Remaining pairs:
    - $S_1(T_2, T_4) = 0.82$
- **Step5:** Replace all null values of attributes within a selected pair
  - $T_2A_2 = T_4A_2$

	A_1	**A_2**	**A_3**	**A_4**	**A_5**
T_1	T_1A_1	T_1A_2	T_1A_3	T_1A_4	T_1A_5
T_2	T_2A_1	***T_4A_2***	T_2A_3	T_2A_4	T_2A_5
T_3	T_3A_1	T_3A_2	T_3A_3	T_3A_4	T_3A_5
T_4	T_4A_1	T_4A_2	T_4A_3	*	T_4A_5
T_5	T_5A_1	T_5A_2	T_5A_3	T_5A_4	T_5A_5
T_6	T_6A_1	*	*	T_6A_4	T_6A_5
T_7	T_7A_1	T_7A_2	T_7A_3	T_7A_4	T_7A_5
T_8	T_8A_1	T_8A_2	*	T_8A_4	T_8A_5
T_9	T_9A_1	T_9A_2	T_9A_3	T_9A_4	T_9A_5
T_{10}	$T_{10}A_1$	$T_{10}A_2$	$T_{10}A_3$	$T_{10}A_4$	$T_{10}A_5$

Table 2. Updated Sample Dataset (SimFiller) After Filling T_2A_2

Updated sample dataset (SimFiller) after filling T_2A_2 is shown in Table 2 where T_2A_2 is replaced with T_4A_2 to fill the missing value of T_2A_2.

2) Next, consider 2nd missing value i.e. T_4A_4 to fill it with SimFiller

- **Step1:** Create pairs of similar data
  - Two pairs are created for T_4
    - $S_1(T_4, T_5)$
    - $S_2(T_4, T_7)$
- **Step2:** Select each pair and allocate it calculated a similarity value
  - $S_1(T_4, T_5) = 0.92$
  - $S_2(T_4, T_7) = 0.87$
- **Step3:** Save each pair having at least one non-null attribute's value
  - No pair omitted
  - Remaining pairs:
    - $S_1(T_4, T_5) = 0.92$
    - $S_2(T_4, T_7) = 0.87$
- **Step4:** Select a pair of highest similarity for each attribute's null value after matching pairs with similarity threshold and with each other
  - $S_2(T_4, T_7)$ omitted because the similarity of $S_1(T_4, T_5) > S_2(T_4, T_7)$
  - Remaining pairs:
    - $S_1(T_4, T_5) = 0.92$
- **Step5:** Replace all null values of attributes within a selected pair
  - There is only one value in pair i.e. T_4A_4 so replacing its value
    - $T_4A_4 = T_5A_4$

	A_1	**A_2**	**A_3**	**A_4**	**A_5**
T_1	T_1A_1	T_1A_2	T_1A_3	T_1A_4	T_1A_5
T_2	T_2A_1	**T_4 A_2**	T_2A_3	T_2A_4	T_2A_5
T_3	T_3A_1	T_3A_2	T_3A_3	T_3A_4	T_3A_5
T_4	T_4A_1	T_4A_2	T_4A_3	***T_5A_4***	T_4A_5
T_5	T_5A_1	T_5A_2	T_5A_3	T_5A_4	T_5A_5
T_6	T_6A_1	*	*	T_6A_4	T_6A_5
T_7	T_7A_1	T_7A_2	T_7A_3	T_7A_4	T_7A_5
T_8	T_8A_1	T_8A_2	*	T_8A_4	T_8A_5
T_9	T_9A_1	T_9A_2	T_9A_3	T_9A_4	T_9A_5
T_{10}	$T_{10}A_1$	$T_{10}A_2$	$T_{10}A_3$	$T_{10}A_4$	$T_{10}A_5$

Table 3. Updated Sample Dataset (SimFiller) After Filling T_4A_4

	A_1	**A_2**	**A_3**	**A_4**	**A_5**
T_1	T_1A_1	T_1A_2	T_1A_3	T_1A_4	T_1A_5
T_2	T_2A_1	**T_4A_2**	T_2A_3	T_2A_4	T_2A_5
T_3	T_3A_1	T_3A_2	T_3A_3	T_3A_4	T_3A_5
T_4	T_4A_1	T_4A_2	T_4A_3	**T_5A_4**	T_4A_5
T_5	T_5A_1	T_5A_2	T_5A_3	T_5A_4	T_5A_5
T_6	T_6A_1	***T_9A_2***	***T_9A_3***	T_6A_4	T_6A_5
T_7	T_7A_1	T_7A_2	T_7A_3	T_7A_4	T_7A_5
T_8	T_8A_1	T_8A_2	***T_5A_3***	T_8A_4	T_8A_5
T_9	T_9A_1	T_9A_2	T_9A_3	T_9A_4	T_9A_5
T_{10}	$T_{10}A_1$	$T_{10}A_2$	$T_{10}A_3$	$T_{10}A_4$	$T_{10}A_5$

Table 4. Updated Sample Dataset (SimFiller) After Filling All Missing Values

Remaining missing values are filled with the same mechanism of SimFiller. Following missing values are filled during the execution of SimFiller:

- $T_2A_2 = T_4A_2$
- $T_4A_4 = T_5A_4$
- $T_6A_2 = T_9A_2$
- $T_6A_3 = T_9A_3$
- $T_8A_3 = T_5A_3$

Updated sample dataset (SimFiller), after filling all possible missing values based on similarity pairs, is shown in Table 4.

3.19. DuDeFiller

DuDeFiller algorithm is an extended form of a SimFiller algorithm which accepts input from the Duplicate Detection Subsystem (DDS) as duplicate record pairs and processes them to fill the missing values of records within pairs. DuDeFiller is duplicate detection based missing values filling algorithm which filters the input duplicate pairs based on similarity threshold, check each pair to verify that at least one record's attribute contains a null value and fills the missing values within each pair.

This algorithm supports the merging of highest similar duplicate record pairs just to fill the missing values of the records' attributes. During the merging missing values of the records are filled and nothing is deleted from the dataset. This technique is useful for data preprocessing to fix the inconsistencies in the dataset related to missing values. Quality of the dataset is increased after passing it from this data cleansing option.

DuDeFiller algorithm fills the missing values of the records and saves the similarity of all duplicate pairs for future calculations. Merge & Fill options developed during the second iteration of Duplicate Fusion Subsystem (DFS) is based on this algorithm. Step-1 to Step-8 are executed on the generated duplicate pairs to fill the missing values of the input dataset. The process continues until are duplicate records are processed and their missing values are filled. The cleaned data is sent back to the Data Cleansing System (DCS) as an output of the Duplicate Fusion Subsystem (DFS) for further processing.

Input duplicate records pass through flowing steps during the DuDeFiller algorithm:

```
Step-1. For each pair {
    Step-2. Pair similarity > similarity threshold {
        Step-3. For each attribute of both records in pair {
                Step-4. Set attribute similarity as "0" if not already set
                Step-5. If only one of the attribute's value is null {
                    Step-6. attribute's similarity >! pair similarity {
                        Step-7. Place non-null attribute's value for null value
                         participant in the pair
                        Step-8. Save attribute's similarity = pair similarity
                    }
                } Step-5. Else {
                    Proceed
                }
            }
    } Step-2. Else {
        Proceed
    }
}
```

3.20. Execution of DuDeFiller

DuDeFiller takes the input dataset and passes it through the eight steps process and generates the cleaned output dataset. Sample dataset containing missing values is considered to fill the missing values using this algorithm as shown in Table 5.

Similarity threshold is fixed at 0.75 for this sample execution and demonstrates that all the records having a similarity greater than 0.75 would be considered for filling the missing values of the sample dataset.

This algorithm supports the merging of duplicate record pairs based on duplicate detection to fill the missing values of the records' attributes. During the merging missing values of the records are filled and nothing is deleted from the dataset.

This technique is useful for data preprocessing to fix the inconsistencies in the dataset related to missing values. Quality of the dataset is increased and the accuracy of the dataset is enhanced after passing the dataset from this data cleansing algorithm.

	A_1	**A_2**	**A_3**	**A_4**	**A_5**
T_1	T_1A_1	T_1A_2	T_1A_3	T_1A_4	T_1A_5
T_2	T_2A_1	*	T_2A_3	T_2A_4	T_2A_5
T_3	T_3A_1	T_3A_2	T_3A_3	T_3A_4	T_3A_5
T_4	T_4A_1	T_4A_2	T_4A_3	*	T_4A_5
T_5	T_5A_1	T_5A_2	T_5A_3	T_5A_4	T_5A_5
T_6	T_6A_1	T_6A_2	T_6A_3	T_6A_4	T_6A_5
T_7	T_7A_1	T_7A_2	T_7A_3	T_7A_4	T_7A_5
T_8	T_8A_1	T_8A_2	*	T_8A_4	T_8A_5
T_9	T_9A_1	T_9A_2	T_9A_3	T_9A_4	T_9A_5
T_{10}	$T_{10}A_1$	$T_{10}A_2$	$T_{10}A_3$	$T_{10}A_4$	$T_{10}A_5$

Table 5. Sample Dataset (DuDeFiller)

Considered sample dataset contains ten records (tuples) represented as T_1 … T_{10} and five attributes represented as A_1… A_5.

After the execution of the Duplicate Detection Subsystem (DDS) on the sample dataset, duplicate pairs are generated and represented as $D_m = (T_n, T_o)$. Where m represents the number allocated to each record pair and n and o represent the record number i.e. $T_1 \ldots T_{10}$

Dataset shown in Table 5 is given as input to the Pre-Processing Subsystem (PPS) which is passed to the Duplicate Detection Subsystem (DDS) after the required preprocessing on it. Duplicate Detection Subsystem (DDS) executes the duplicate detection algorithms on the preprocessed input dataset and generates the duplicate records pairs given below:

- $D_1(T_1, T_2)$
- $D_2(T_1, T_3)$
- $D_3(T_2, T_4)$
- $D_4(T_2, T_5)$
- $D_5(T_3, T_4)$
- $D_6(T_8, T_9)$

The calculated similarity of each duplicate pair received from the Duplicate Detection Subsystem (DDS) is assigned to each duplicate pair and is given as input to the DuDeFiller algorithm for filling the missing values of the dataset.

- $D_1(T_1, T_2) = 0.85$
- $D_2(T_1, T_3) = 0.80$
- $D_3(T_2, T_4) = 0.65$
- $D_4(T_2, T_5) = 0.87$
- $D_5(T_3, T_4) = 0.86$
- $D_6(T_8, T_9) = 0.80$

All duplicate pairs are processed by DuDeFiller algorithm from the Step-1 to Step-8 and missing values of the dataset are filled from the pair having highest similarity.

Execution of DuDeFiller for D_1

- ☑ Step-1: Selected pair $D_1(T_1, T_2)$
- ☑ Step-2: 0.85 > 0.75 (passed)
- ☑ Step-3: Selected attributes of T_1 and T_2

- ☑ Step-4: Assigned mutual similarity for all attributes of T_1 and T_2 = 0 as not already saved
- ☑ Step-5: Attribute A_2 for T_2 is null (passed)
- ☑ Step-6: (T_2A_2 similarity) 0 >! 0.85 (Pair similarity)
- ☑ Step-7: $T_2A_2 = T_1A_2$
- ☑ Step-8: T_2A_2 similarity = 0.85

Sample dataset is updated after the execution of DuDeFiller for the pair D_1 as shown in Table 6 where null value T_2A_2 is filled with T_1A_2.

	A_1	**A_2**	**A_3**	**A_4**	**A_5**
T_1	T_1A_1	T_1A_2	T_1A_3	T_1A_4	T_1A_5
T_2	T_2A_1	***T_1A_2***	T_2A_3	T_2A_4	T_2A_5
T_3	T_3A_1	T_3A_2	T_3A_3	T_3A_4	T_3A_5
T_4	T_4A_1	T_4A_2	T_4A_3	*****	T_4A_5
T_5	T_5A_1	T_5A_2	T_5A_3	T_5A_4	T_5A_5
T_6	T_6A_1	T_6A_2	T_6A_3	T_6A_4	T_6A_5
T_7	T_7A_1	T_7A_2	T_7A_3	T_7A_4	T_7A_5
T_8	T_8A_1	T_8A_2	*	T_8A_4	T_8A_5
T_9	T_9A_1	T_9A_2	T_9A_3	T_9A_4	T_9A_5
T_{10}	$T_{10}A_1$	$T_{10}A_2$	$T_{10}A_3$	$T_{10}A_4$	$T_{10}A_5$

Table 6. Updated Sample Dataset (DuDeFiller) After Processing of D_1

Execution of DuDeFiller for D_2

- ☑ Step-1: Selected pair $D_2(T_1, T_3)$
- ☑ Step-2: 0.80 > 0.75 (passed)
- ☑ Step-3: Selected attributes of T_1 and T_3
- ☑ Step-4: Assigned mutual similarity for all attributes of T_1 and T_3 = 0 as not already saved
- ☒ Step-5: None of the attributes is null for T_1 and T_3 (failed)

DuDeFiller algorithm exists from Step-5 for D_2 as none of the attributes for T_1 and T_3 has a null value and sample dataset is not updated.

Execution of DuDeFiller for D_3

- ☑ Step-1: Selected pair $D_3(T_2, T_4)$
- ☒ Step-2: 0.65 > 0.75 (failed)

DuDeFiller algorithm exits from Step-2 for D_3 as pair similarity is less than the similarity threshold and sample dataset is not updated.

Execution of DuDeFiller for D_4

- ☑ Step-1: Selected pair $D_4(T_2, T_5)$
- ☑ Step-2: 0.87 > 0.75 (passed)
- ☑ Step-3: Selected attributes of T_2 and T_5
- ☑ Step-4: Assigned mutual similarity for all attributes of T_5 = 0 but not for and T_2 as it is already saved as T_2A_2 similarity = 0.85
- ☑ Step-5: Attribute A_2 for T_2 is initially null (passed)
- ☑ Step-6: (T_2A_2 similarity) 0.85 >! 0.87 (Pair similarity)
- ☑ Step-7: $T_2A_2 = T_5A_2$ (again updated)
- ☑ Step-8: T_2A_2 similarity = 0.87 (again updated)

Sample dataset is again updated after the execution of DuDeFiller for the pair D_4 as shown in Table 7 where T_2A_2 is filled with T_5A_2.

	A_1	**A_2**	**A_3**	**A_4**	**A_5**
T_1	T_1A_1	T_1A_2	T_1A_3	T_1A_4	T_1A_5
T_2	T_2A_1	***T_5A_2***	T_2A_3	T_2A_4	T_2A_5
T_3	T_3A_1	T_3A_2	T_3A_3	T_3A_4	T_3A_5
T_4	T_4A_1	T_4A_2	T_4A_3	*****	T_4A_5
T_5	T_5A_1	T_5A_2	T_5A_3	T_5A_4	T_5A_5
T_6	T_6A_1	T_6A_2	T_6A_3	T_6A_4	T_6A_5
T_7	T_7A_1	T_7A_2	T_7A_3	T_7A_4	T_7A_5
T_8	T_8A_1	T_8A_2	*	T_8A_4	T_8A_5
T_9	T_9A_1	T_9A_2	T_9A_3	T_9A_4	T_9A_5
T_{10}	$T_{10}A_1$	$T_{10}A_2$	$T_{10}A_3$	$T_{10}A_4$	$T_{10}A_5$

Table 7. Updated Sample Dataset (DuDeFiller) After Processing of D_4

Execution of DuDeFiller for D_5

- ☑ Step-1: Selected pair $D_5(T_3, T_4)$
- ☑ Step-2: 0.86 > 0.75 (passed)
- ☑ Step-3: Selected attributes of T_3 and T_4
- ☑ Step-4: Assigned mutual similarity for all attributes of T_3 and T_4 = 0 as not already saved
- ☑ Step-5: Attribute A_4 for T_4 is null (passed)
- ☑ Step-6: (T_4A_4 similarity) 0 >! 0.86 (Pair similarity)
- ☑ Step-7: $T_4A_4 = T_3A_4$
- ☑ Step-8: T_4A_4 similarity = 0.86

Sample dataset is again updated after the execution of DuDeFiller for the pair D_5 as shown in Table 8 where null value T_4A_4 is filled with T_3A_4.

	A_1	**A_2**	**A_3**	**A_4**	**A_5**
T_1	T_1A_1	T_1A_2	T_1A_3	T_1A_4	T_1A_5
T_2	T_2A_1	**T_5A_2**	T_2A_3	T_2A_4	T_2A_5
T_3	T_3A_1	T_3A_2	T_3A_3	T_3A_4	T_3A_5
T_4	T_4A_1	T_4A_2	T_4A_3	***T_3A_4***	T_4A_5
T_5	T_5A_1	T_5A_2	T_5A_3	T_5A_4	T_5A_5
T_6	T_6A_1	T_6A_2	T_6A_3	T_6A_4	T_6A_5
T_7	T_7A_1	T_7A_2	T_7A_3	T_7A_4	T_7A_5
T_8	T_8A_1	T_8A_2	*****	T_8A_4	T_8A_5
T_9	T_9A_1	T_9A_2	T_9A_3	T_9A_4	T_9A_5
T_{10}	$T_{10}A_1$	$T_{10}A_2$	$T_{10}A_3$	$T_{10}A_4$	$T_{10}A_5$

Table 8. Updated Sample Dataset (DuDeFiller) After Processing of D_5

Execution of DuDeFiller for D_6

- ☑ Step-1: Selected pair $D_6(T_8, T_9)$
- ☑ Step-2: 0.80 > 0.75 (passed)
- ☑ Step-3: Selected attributes of T_8 and T_9

- ☑ Step-4: Assigned mutual similarity for all attributes of T_8 and T_9 = 0 as not already saved
- ☑ Step-5: Attribute A_3 for T_8 is null (passed)
- ☑ Step-6: (T_8A_3 similarity) 0 >! 0.80 (Pair similarity)
- ☑ Step-7: $T_8A_3 = T_9A_3$
- ☑ Step-8: T_8A_3 similarity = 0.80

Sample dataset is again updated after the execution of DuDeFiller for the pair D_6 as shown in Table 9 where null value T_8A_3 is filled with T_9A_3.

	A_1	**A_2**	**A_3**	**A_4**	**A_5**
T_1	T_1A_1	T_1A_2	T_1A_3	T_1A_4	T_1A_5
T_2	T_2A_1	**T_5A_2**	T_2A_3	T_2A_4	T_2A_5
T_3	T_3A_1	T_3A_2	T_3A_3	T_3A_4	T_3A_5
T_4	T_4A_1	T_4A_2	T_4A_3	**T_3A_4**	T_4A_5
T_5	T_5A_1	T_5A_2	T_5A_3	T_5A_4	T_5A_5
T_6	T_6A_1	T_6A_2	T_6A_3	T_6A_4	T_6A_5
T_7	T_7A_1	T_7A_2	T_7A_3	T_7A_4	T_7A_5
T_8	T_8A_1	T_8A_2	***T_9A_3***	T_8A_4	T_8A_5
T_9	T_9A_1	T_9A_2	T_9A_3	T_9A_4	T_9A_5
T_{10}	$T_{10}A_1$	$T_{10}A_2$	$T_{10}A_3$	$T_{10}A_4$	$T_{10}A_5$

Table 9. Updated Sample Dataset (DuDeFiller) After Processing of D_6

DuDeFiller algorithm exists after the execution of the last duplicate pair and tries to fill all the missing values found in the input duplicate pairs.

3.21. DuDeFuse

DuDeFuse is duplicate detection and fusion algorithm which accepts input from the Duplicate Detection Subsystem as duplicate record pairs. DuDeFuse calculates the similarity and occurrence of each possible value of an attribute for each of the records within the duplicate pair. This algorithm extracts each record's every attribute's value having maximum similarity sum and the value of having maximum occurrence.

Input duplicate records pass through flowing steps during the DuDeFuse algorithm:

```
Step-1. For each pair {
  Step-2. Pair similarity > similarity threshold {
    Step-3. For each attribute of both records in pair {
      Step-4. If attribute's value is not null {
        Step-5. If attribute's value not already saved {
          Step-6. Save its value
          Step-7. Set occurrence = 1
          Step-8. Save attribute's similarity = pair similarity
        } Step-5. Else {
          Step-9. Increment occurrence = 1
          Step-10. Add pair similarity to saved similarity
        }
      } Step-4. Else {
        Proceed
      }
    }
  } Step-2. Else {
    Proceed
  }
}
```

Duplicate merging is implemented based on the input provided by the user for the duplicate fusion options. This algorithm supports the Maximum Similarity Sum and Maximum Occurrences options of the duplicate fusion to merge the duplicate records.

3.22. Post-Operation Subsystem (POS)

Post-Operation Subsystem (POS) deals with the activities executed to create the output file after performing a cleansing task on it during the Duplicate Fusion Subsystem (DFS). DCS passes the output of the Data Fusion Subsystem (DFS) to the Post-Operation Subsystem (POS) in the form of fused duplicates which is further processed and written to the file for the user.

Post-Operation Subsystem (POS) is further divided into the following four main modules as shown in Fig. 14.

- Create an output file
    - Create a CSV file (having .csv extension) with a proper name derived from the input file and this file is created in a similar directory from where the data was imported.
- Write Attributes names
    - Attributes names those were extracted from the input file in the Pre-Processing Subsystem are written in the output file according to the CSV format.
- Write Fused/Cleaned records
    - Fused or Cleaned records those were received from the Duplicate Fusion Subsystem based on the selected user operation are written one by one into the output file according to the CSV format.
- Open-File
    - After writing the file in the system directory it is opened for the user.

The output file is created by the POS to store the output of the whole process carried out from the Pre-Processing Subsystem (PPS) to the Post-Operation Subsystem (POS) and it contains the following information:

- Attributes header
- Cleaned records after performing one of the below operations
    - Filling missing values
    - Fusing duplicate records

Data Cleansing System (DCS) is developed in two iterations by applying Solo Iterative Process (SIP). During the first iteration, basic functionalities are implemented to create a preliminary end to end system. During the second iteration, advanced functionalities are added

to the system by extending the existing developed functionalities. Process flow details of each submodule with respect to iteration are discussed in the next sections.

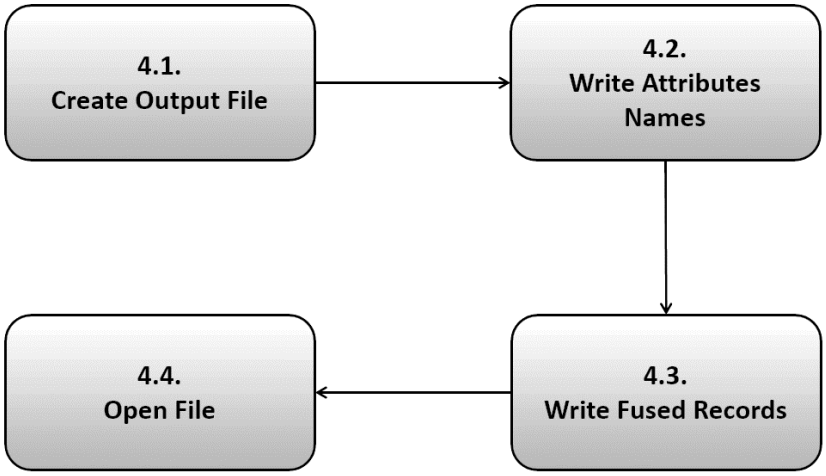

Fig. 14. Post-Operation Subsystem (POS)

3.23. POS Flowchart

The sequence of steps performed by the developed Post-Operation Subsystem (POS) is shown in Fig. 15. POS is developed completely in the single iteration because it is just related to file writing and post-processing functionality.

Details of the steps performed by the developed subsystem are given below:

- Create an output file
- Read cleaned data
- Check no of cleaned records
- Write attributes if no of cleaned records > 0
- Select each cleaned record one by one
- Write each selected cleaned record to the created file
- Save the output file
- Open the output file after writing all cleaned records

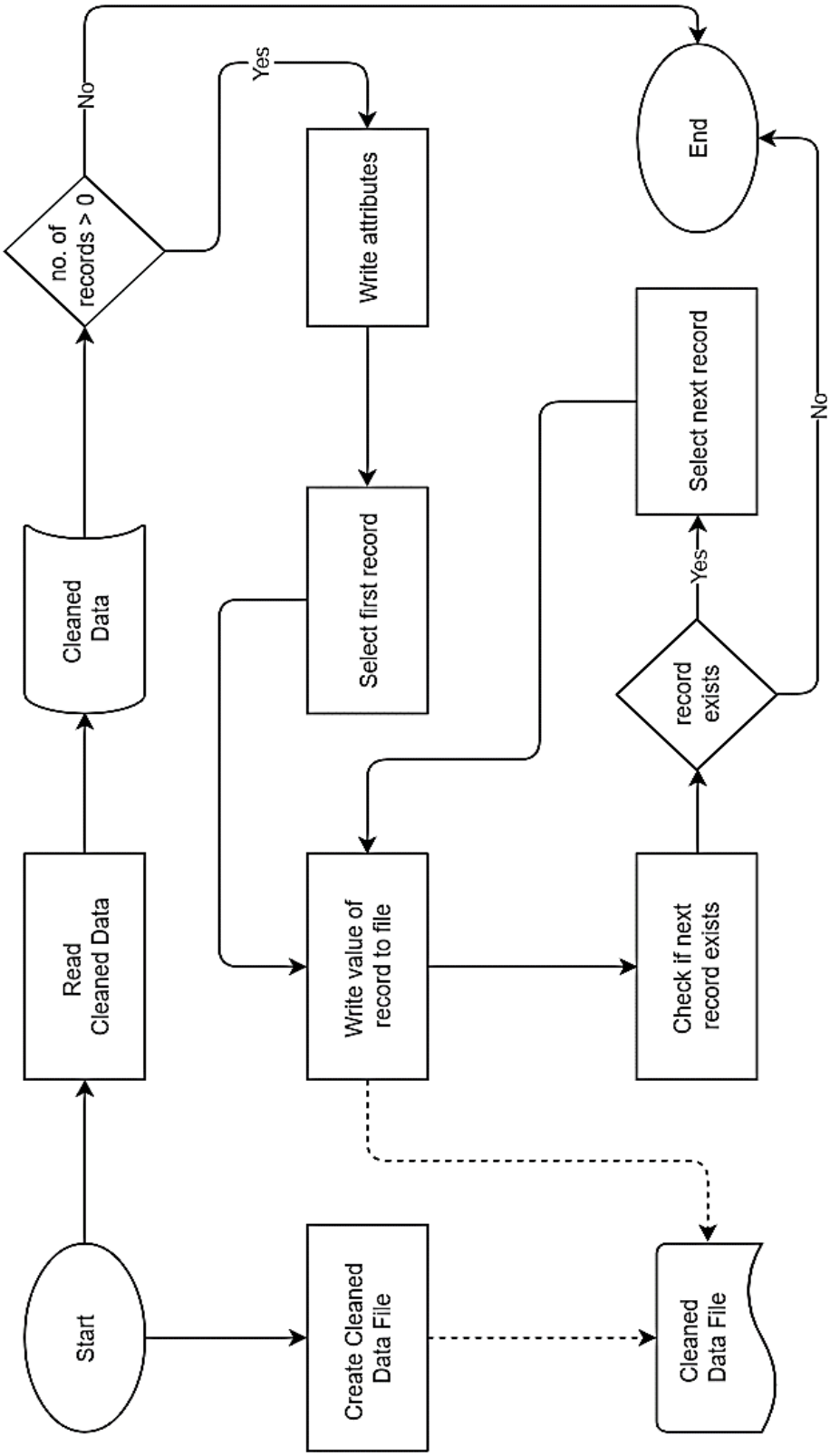

Fig. 15. Post-Operation Subsystem (POS) Flowchart

3.24. Operation of Data Cleansing System

A data quality improvement system, named as ***Data Cleansing System (DCS)*** is based on duplicate detection mechanism during which record having some similarities (duplication) with each other are extracted in the form of pairs named as duplicate pairs. The extracted duplicate pairs are utilized to take the duplicate merging decisions by the system. Moreover, this system also utilizes these duplicates pairs to fill the missing values within the pairs. Data Cleansing System (DCS) provides the following four major functions related to data cleansing:

- Import and Preprocess (Pre-Processing Subsystem) / (PPS)
- Find Duplicates (Duplicate Detection Subsystem) / (DDS)
- Fill Missing Values (Duplicate Fusion Subsystem) / (DFS)
- Fuse Duplicates (Duplicate Fusion Subsystem) / (DFS)

Following are the instructions for the user to efficiently operate the DCS.

1) Import and Preprocess (Pre-Processing Subsystem) / (PPS)

Pre-Processing Subsystem deals with the preliminary activities executed on the input file before performing any data cleansing task on it. Following user interface is created with "Import and Preprocess (PPS)" button, as shown in Fig. 16, to select and load the CSV file in the subsystem.

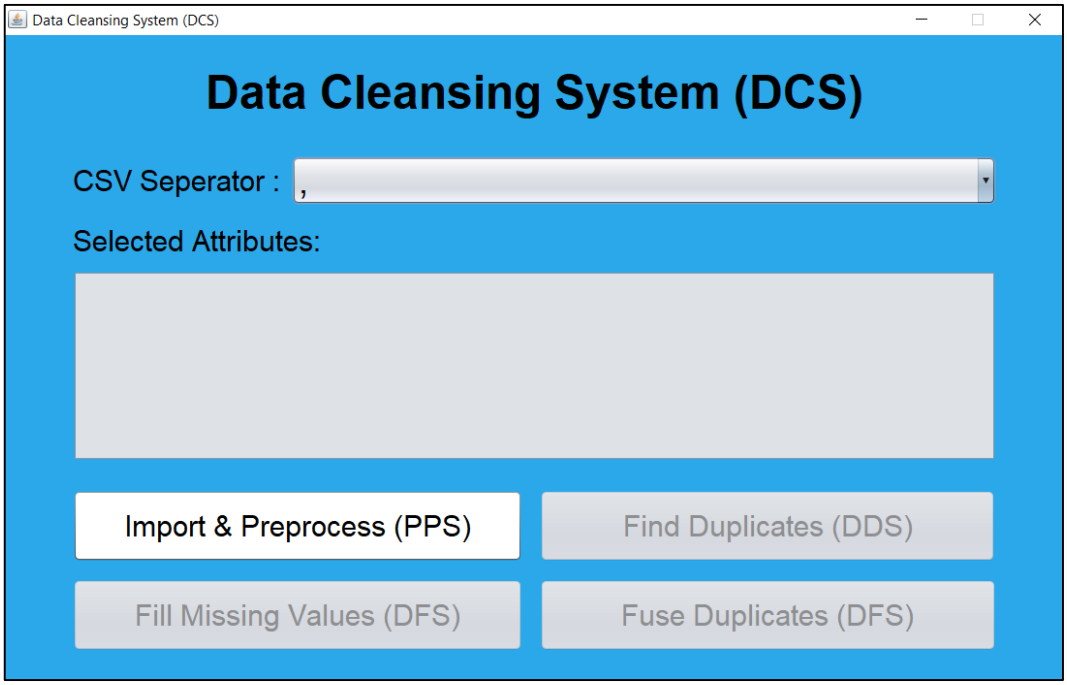

Fig. 16. Data Cleansing System (Screen)

CSV separator selection is customized so that user can select the dataset's CSV separator either "**,**" or "**;**" as shown in Fig, 17.

Fig. 17. CSV Separator Selection (PPS)

On clicking the "Import and Preprocess (PPS)" button, PPS asks the user to select and upload the raw dataset CSV file into the subsystem as shown in Fig. 18.

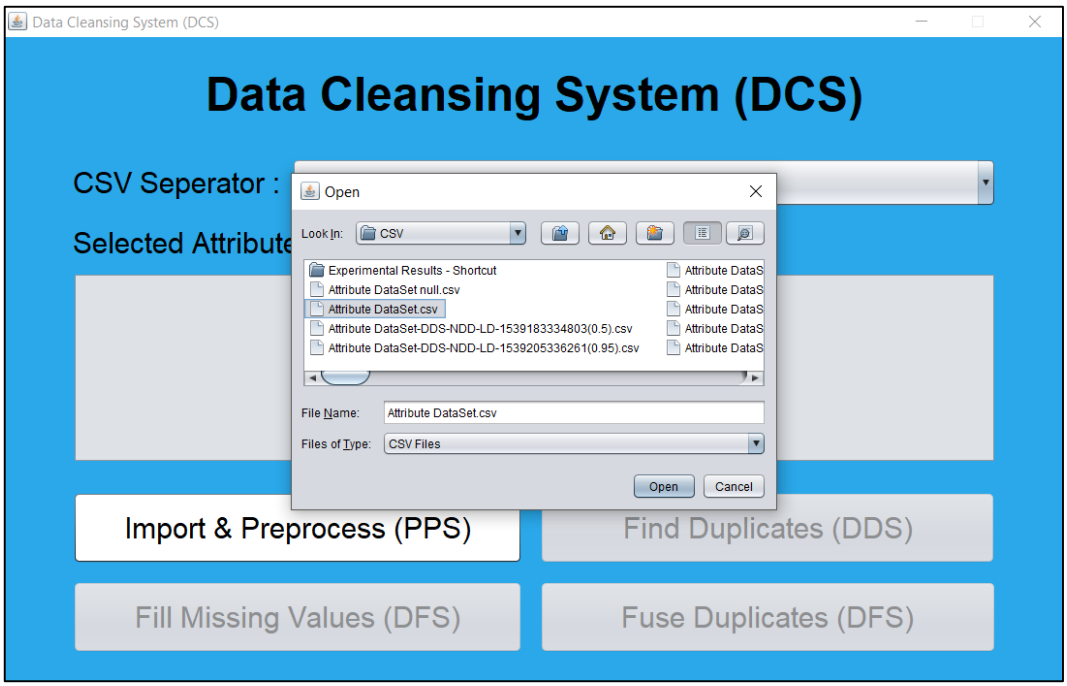

Fig. 18. Select and Upload File (PPS)

The user selects and uploads the file. After successful loading, PPS reads the input dataset from CSV file, extracts the attribute list from it and reads data for all the extracted attributes of the file.

PPS shows the instructions to the users which attributes are recommended to select and which are not recommended to select as shown in Fig. 19.

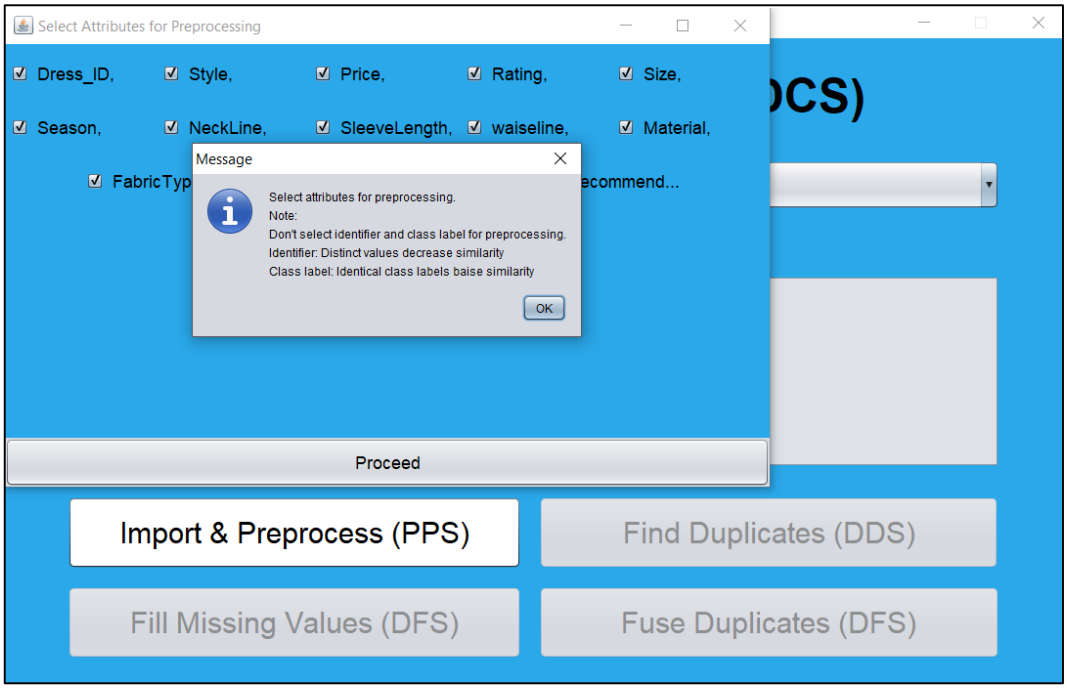

Fig. 19. Instructions to Select Attributes (PPS)

The subsystem asks the user to select the required attributes from the extracted list of attributes. Initially, all attributes are selected by the subsystem as shown in Fig. 20.

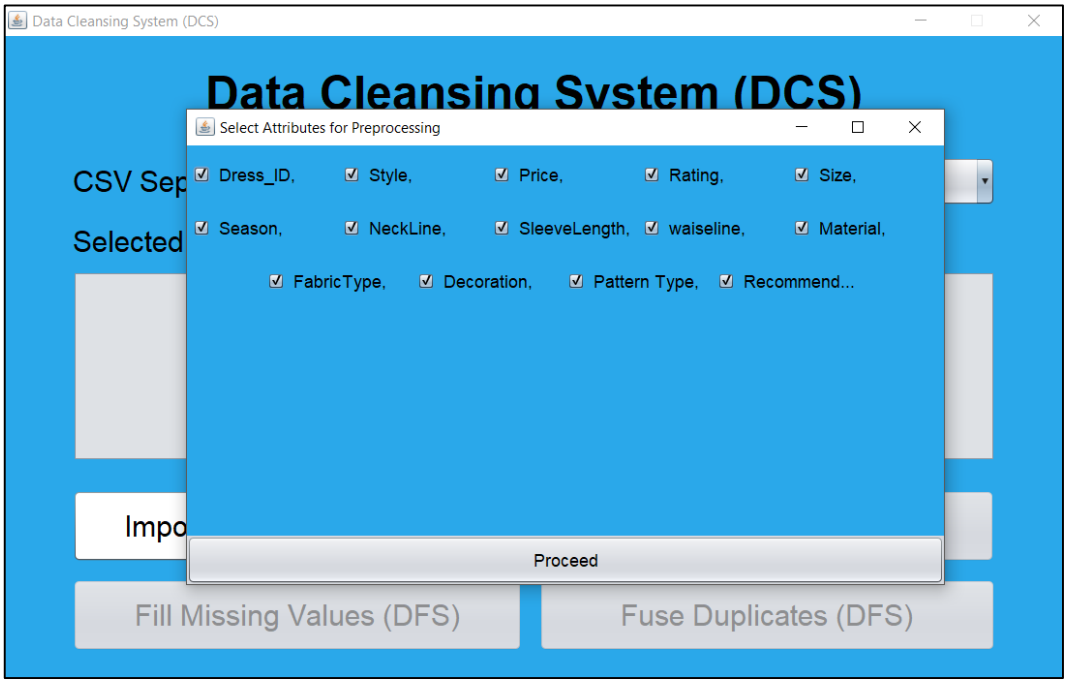

Fig. 20. Attributes Selection (PPS)

The selected attributes are used to extract the required data from the dataset which is returned to the main system for further processing. PPS enables the “Find Duplicates (DDS)” button, “Fill Missing Values (DFS)” button and “Fuse Duplicates (DFS)” button

after successful selection of attributes. It also loads the selected attributes in the selected attributes box given in Fig. 21.

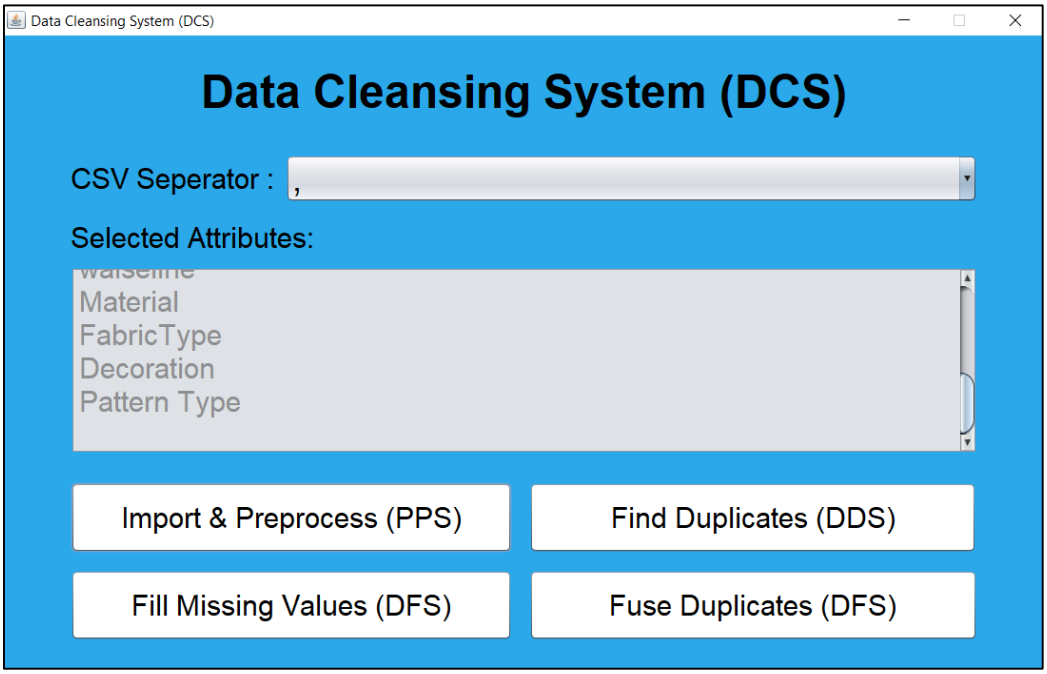

Fig. 21. Selected Attributes (PPS)

2) Find Duplicates (Duplicate Detection Subsystem) / (DDS)

After clicking the "Find Duplicates (DDS)" button on the screen shown in Fig. 22, the system takes the user to the Duplicate Detection Subsystem.

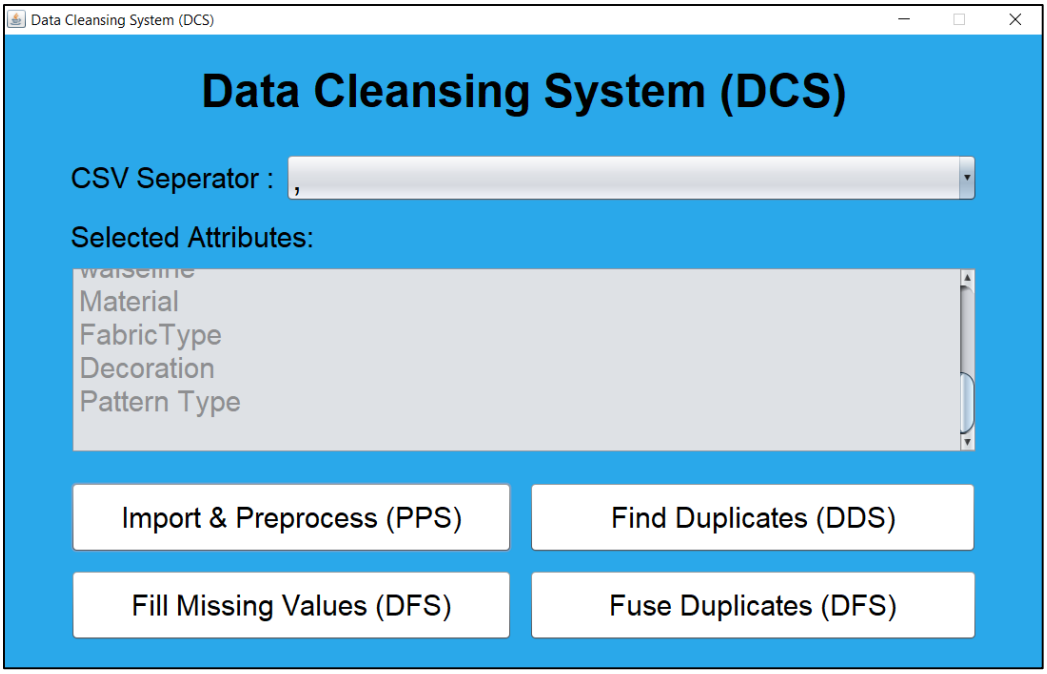

Fig. 22. Selected Attributes (DDS)

Duplicate Detection Subsystem asks the user to pick the algorithm for duplicate detection and generates possible duplicate pairs after comparing records of the given dataset. Duplicate Detection Subsystem (DDS) interface is shown in Fig. 23.

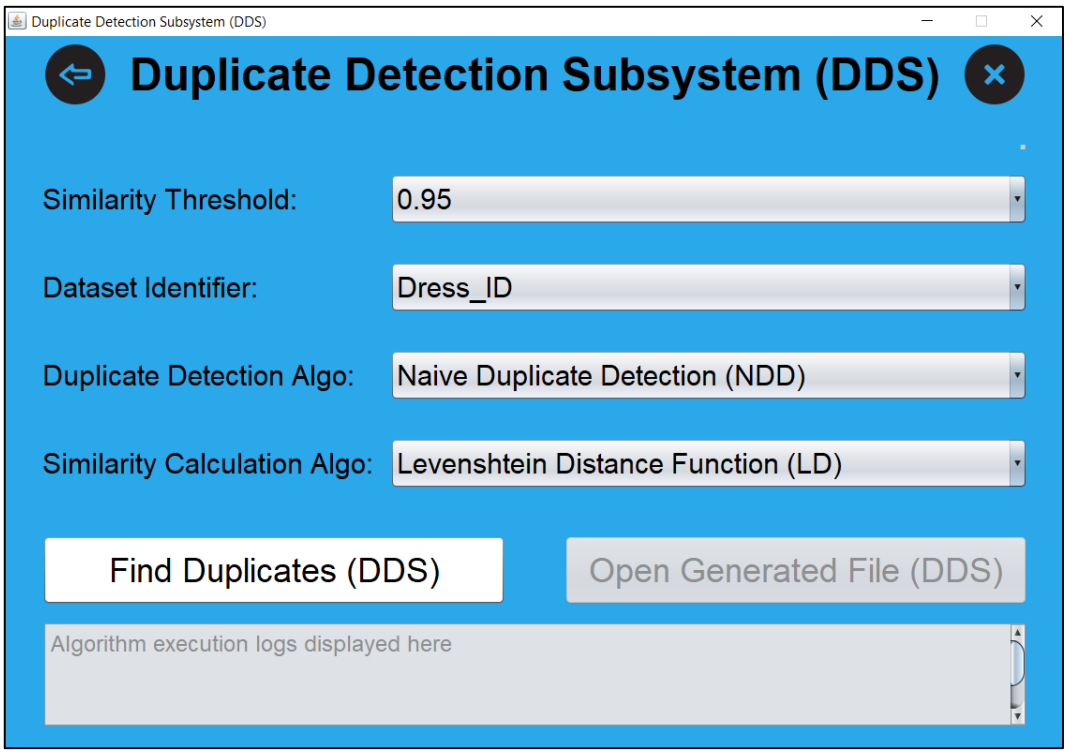

Fig. 23. Duplicate Detection Subsystem

Similarity threshold selection is customized so that user can select the appropriate threshold from the options shown in Fig. 24.

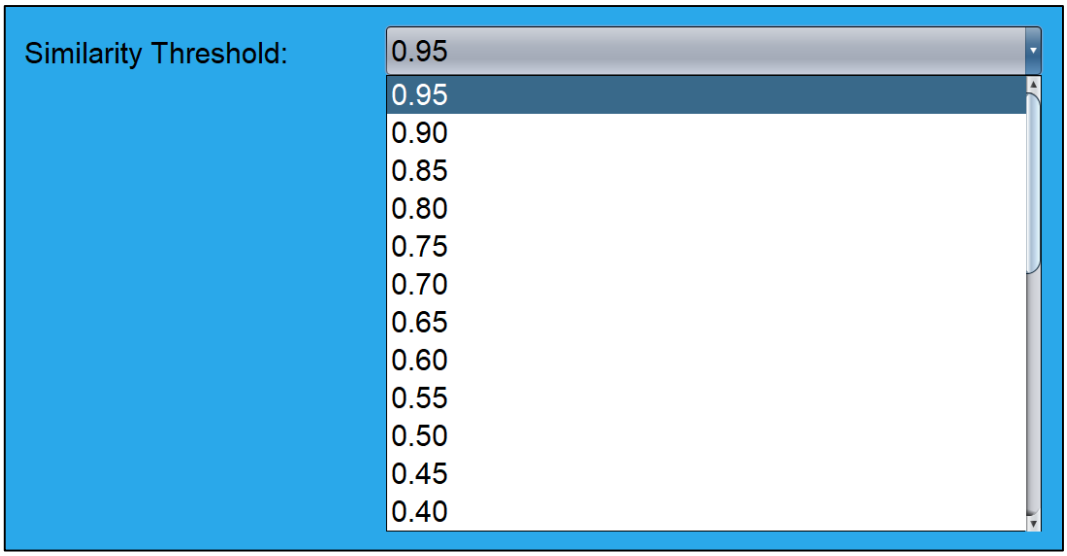

Fig. 24. Similarity Threshold Selection (DDS)

Dataset identifier is customized so that user can select the appropriate identifier from the list of attributes extracted from the imported data as shown in Fig. 25.

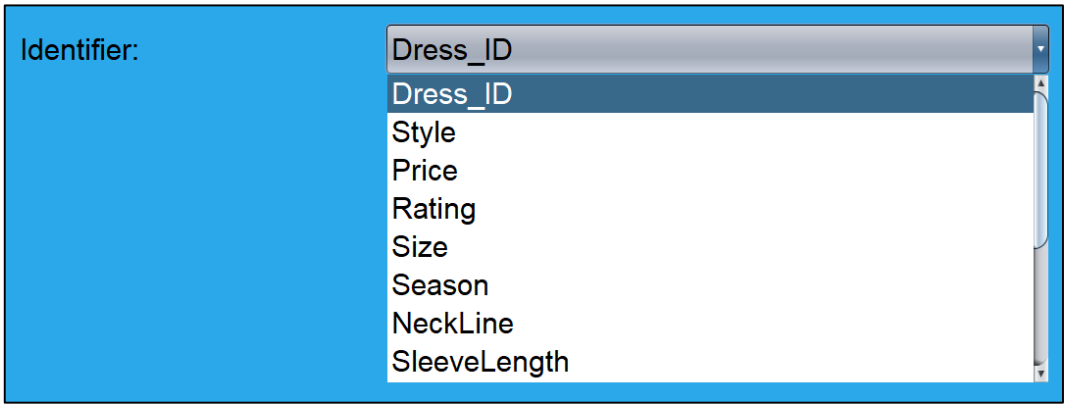

Fig. 25. Dataset Identifier Selection (DDS)

Duplicate detection algorithm selection is customized so that the user can select the appropriate duplicate detection algorithm from the options shown in Fig. 26.

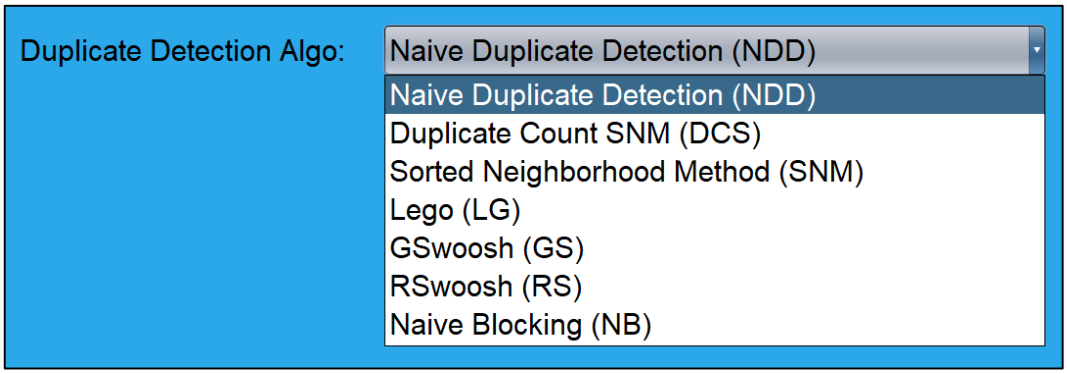

Fig. 26. Duplicate Detection Algorithm Selection (DDS)

Similarity calculation algorithm selection is customized so that the user can select the appropriate similarity calculation algorithm from the options given in the Fig.27.

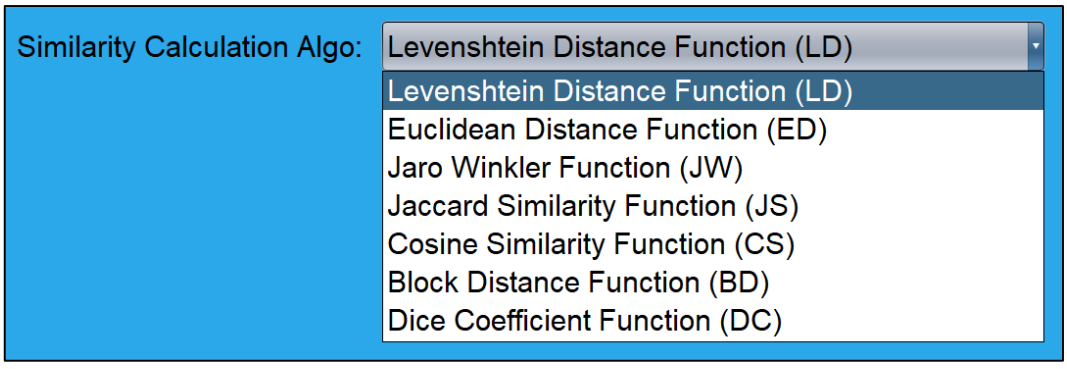

Fig. 27. Similarity Calculation Algorithm Selection (DDS)

After clicking the “Find Duplicates (DDS)” button, the subsystem starts finding the duplicate records from the input dataset as shown in Fig. 28.

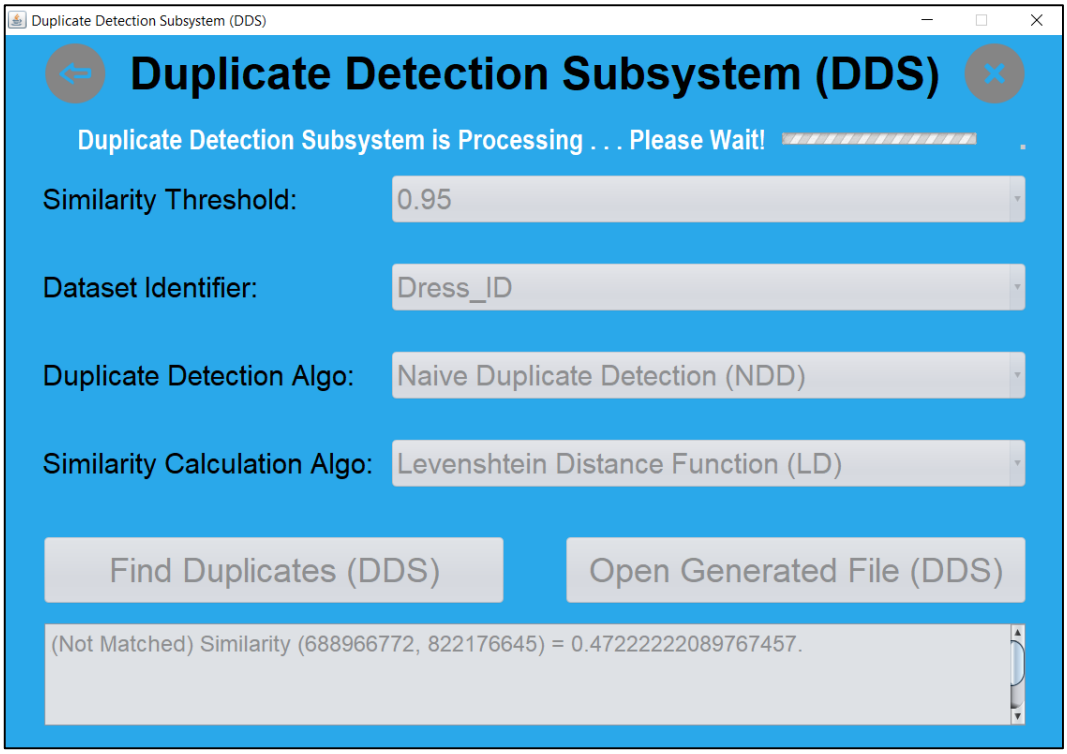

Fig. 28. DDS Processing

A success message is displayed after finding and writing the duplicate records from the input data set to the output file as shown in Fig. 29 and "Open Generated File (DDS)" button is also enabled to open the generated output file of duplicate records.

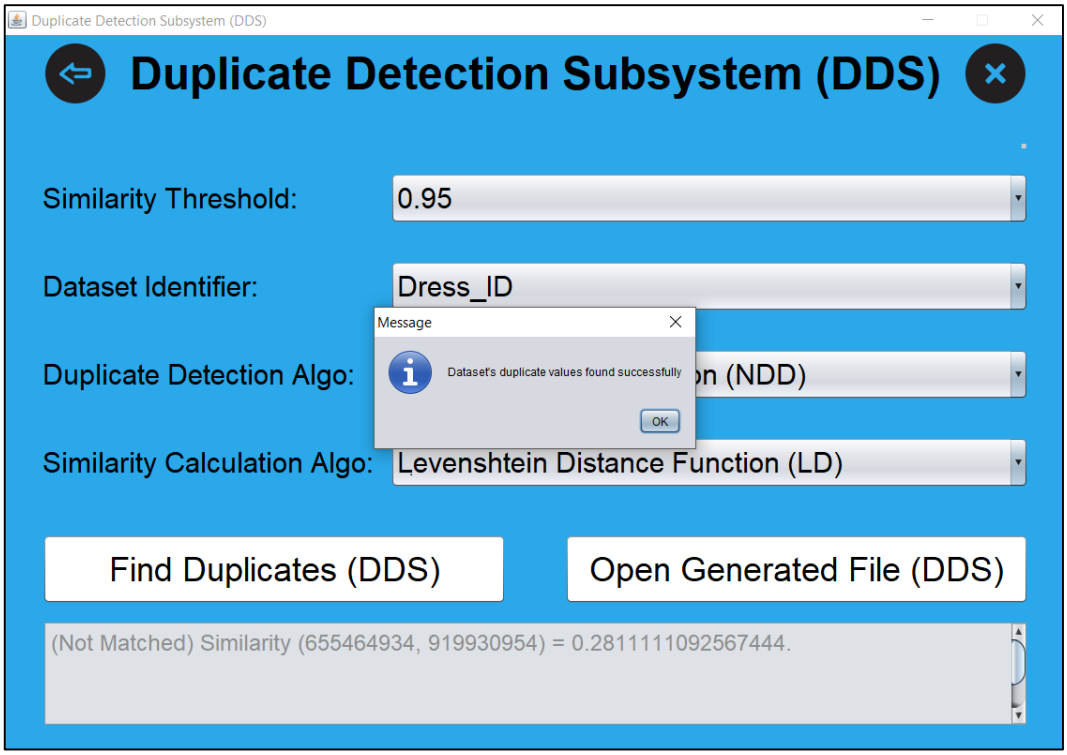

Fig. 29. DDS Processing Completed

3) Fill Missing Values (Duplicate Fusion Subsystem) / (DFS)

After clicking the "Fill Missing Values (DFS)" button on the screen shown in Fig. 30, the system takes the user to the DFS for Missing Values (DFS-MV).

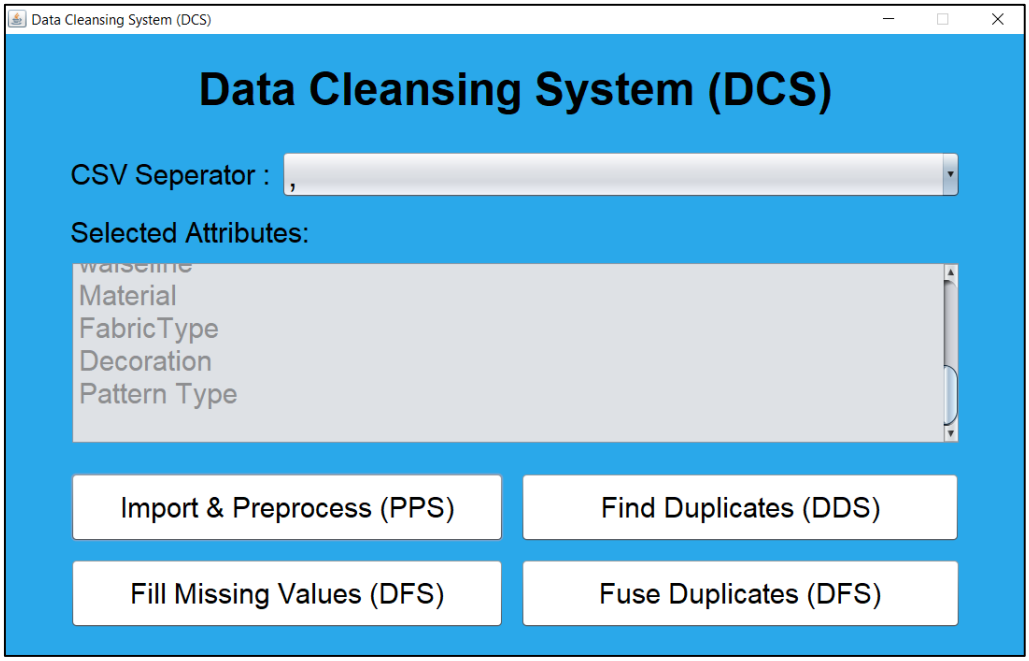

Fig. 30. Selected Attributes (DFS - MV)

Duplicate Fusion Subsystem for Missing Values (DFS-MV) supports the merging of highest similar duplicate record pairs just to fill the missing values of the records' attributes and nothing is deleted from the dataset. DFS-MV interface is shown in Fig. 31.

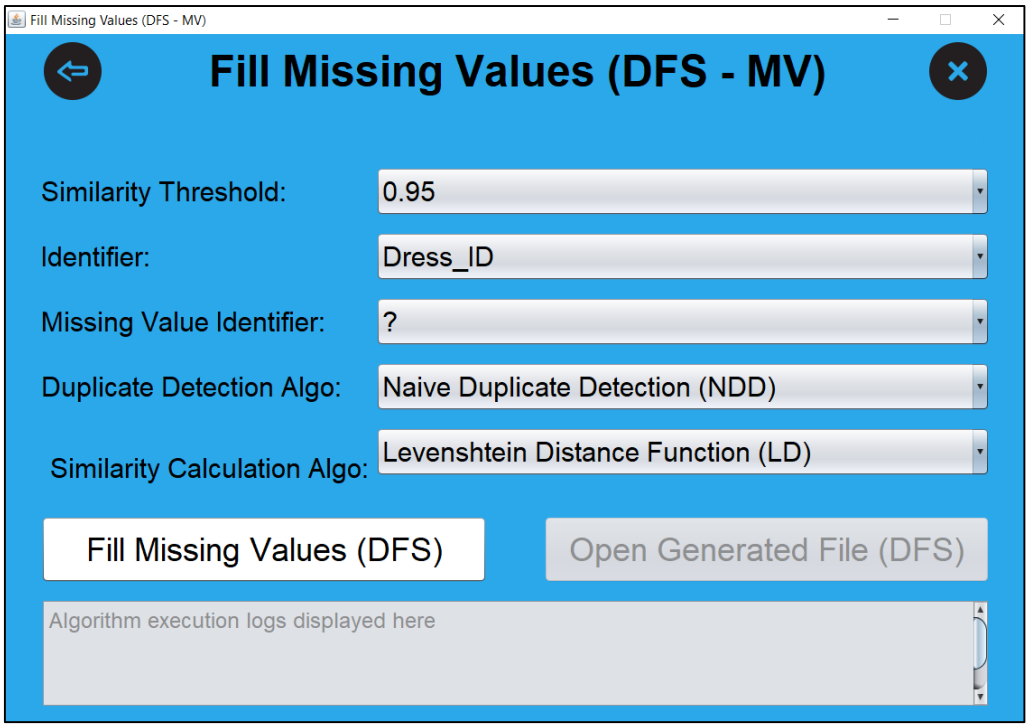

Fig. 31. Duplicate Fusion Subsystem for Missing Values (DFS-MV)

Similarity threshold selection is customized for the Duplicate Fusion Subsystem for Missing Values (DFS-MV) so that user can select the appropriate threshold from the options given in Fig. 32.

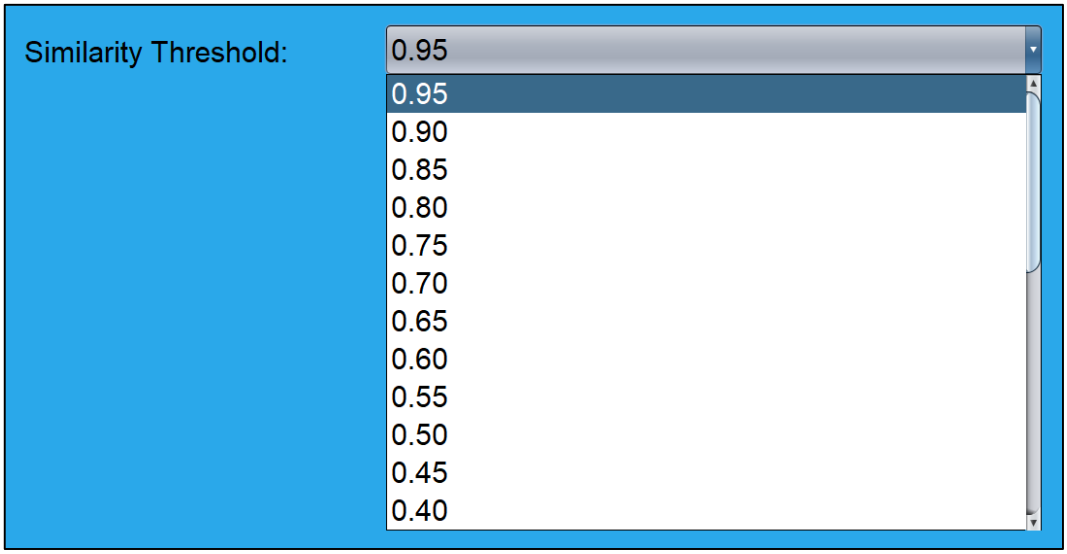

Fig. 32. Similarity Threshold Selection (DFS-MV)

Dataset identifier is customized for the Duplicate Fusion Subsystem for Missing Values (DFS-MV) so that user can select the appropriate identifier from the list of attributes extracted from the imported data as shown in Fig. 33.

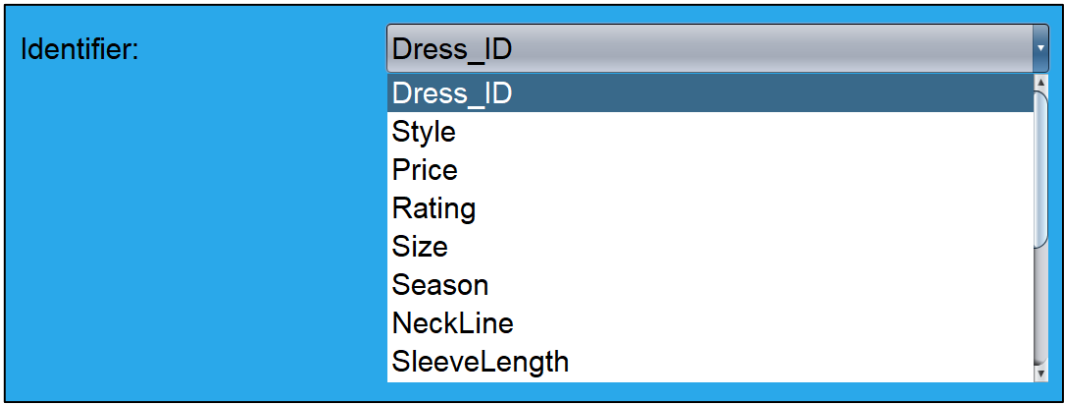

Fig. 33. Dataset Identifier Selection (DFS-MV)

Missing value representation selection is customized for the Duplicate Fusion Subsystem for Missing Values (DFS-MV) so that user can select the "?" or "null" as the missing value representation according to the input data as shown in Fig. 34.

Fig. 34. Missing Value Representation Selection (DFS-MV)

Duplicate detection algorithm selection is customized for the Duplicate Fusion Subsystem for Missing Values (DFS-MV) so that user can select the appropriate duplicate detection algorithm from the options given in Fig. 35.

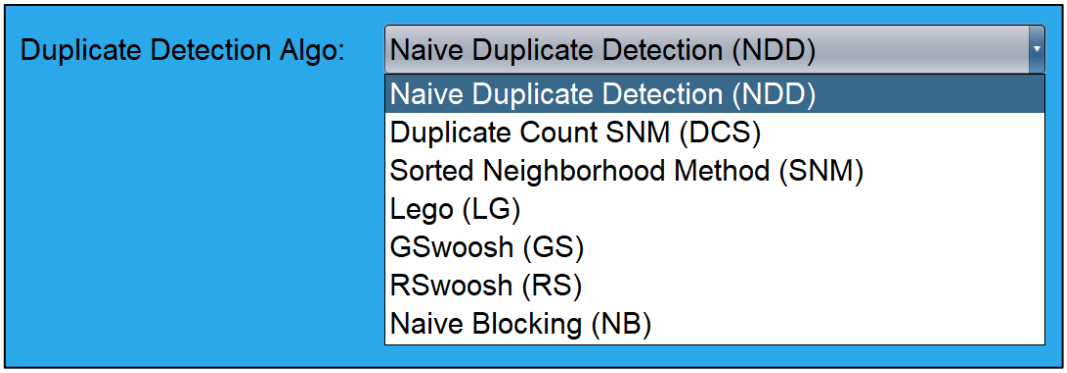

Fig. 35. Duplicate Detection Algorithm Selection (DFS-MV)

Similarity calculation algorithm selection is customized for the Duplicate Fusion Subsystem for Missing Values (DFS-MV) so that user can select the appropriate similarity calculation algorithm from the options given in Fig. 36.

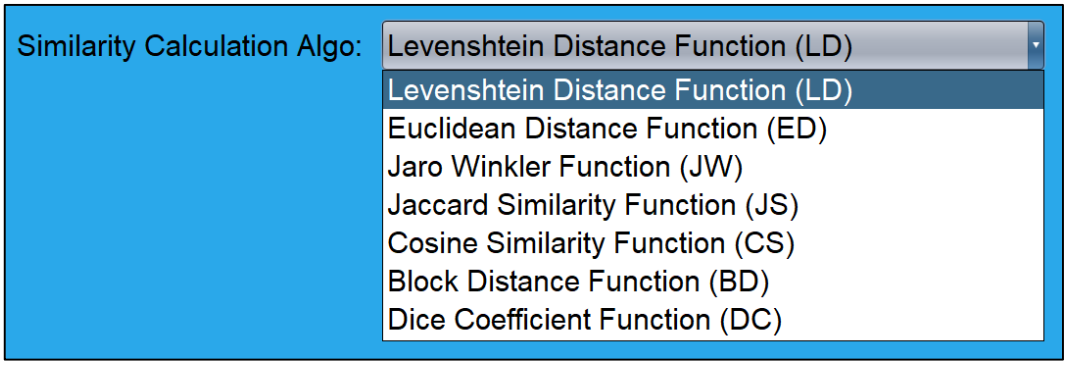

Fig. 36. Similarity Calculation Algorithm Selection (DFS-MV)

After clicking the "Fill Missing Values (DFS)" button, subsystem starts filling the missing values of the input dataset by merging of highest similar duplicate record pairs just to fill the missing values of the records' attributes and nothing is deleted from the dataset as shown in the Fig. 37.

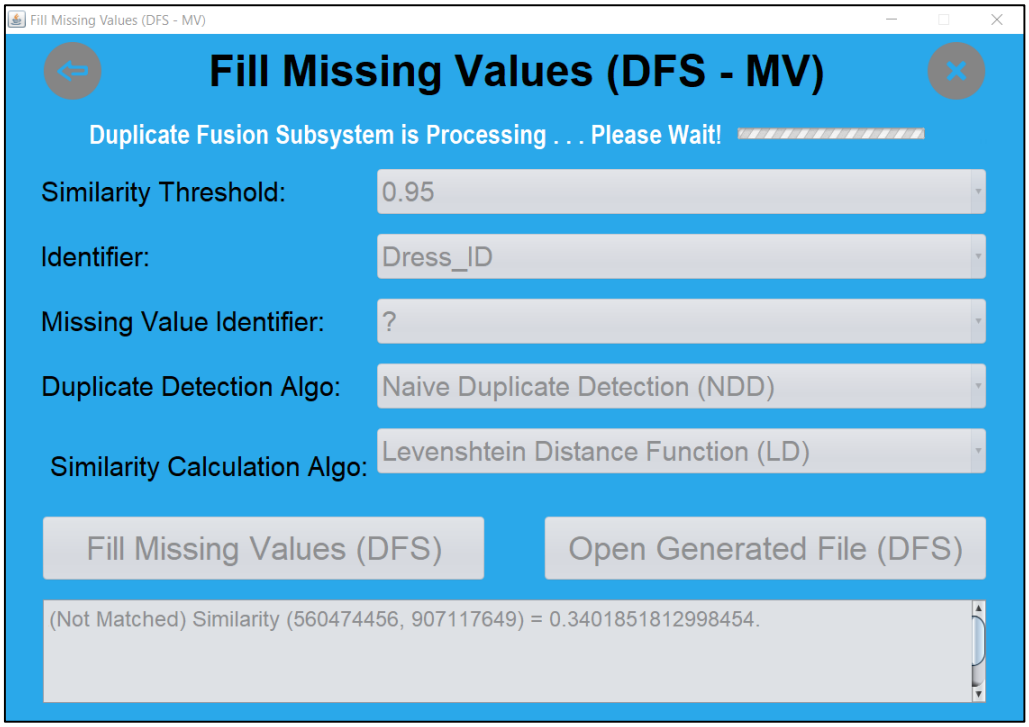

Fig. 37. DFS-MV Processing

A success message is displayed after filling the missing values and writing the cleaned records of the input data set to the output file as shown in Fig. 38. “Open Generated File (DFS)” button is also enabled to open and show the generated output file with the information of cleaned records.

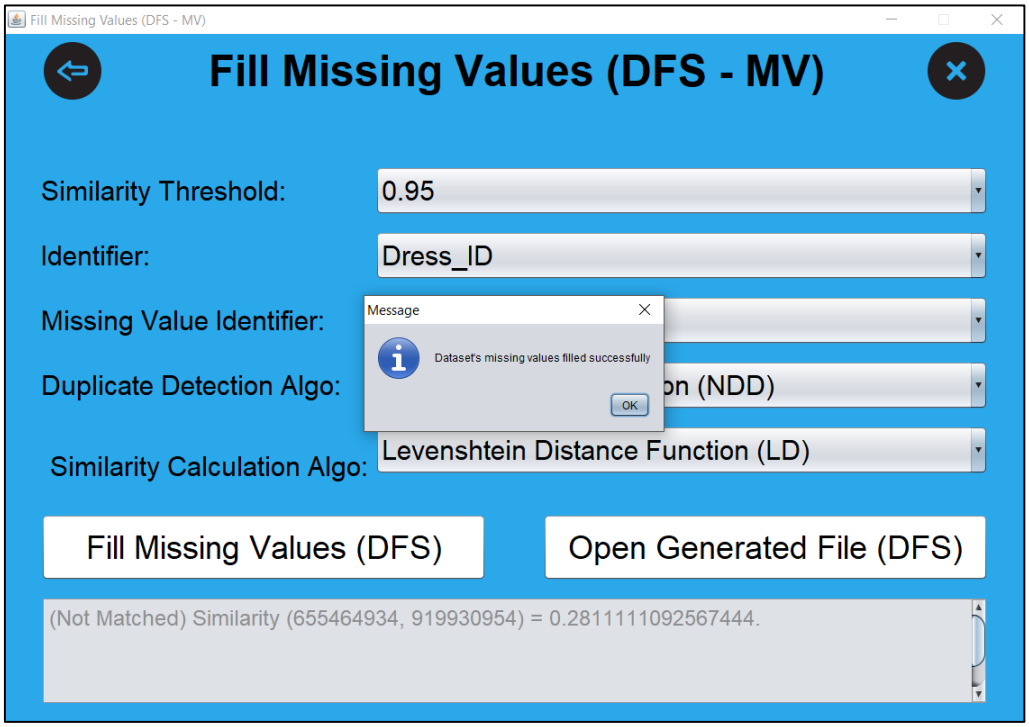

Fig. 38. DFS-MV Processing Completed

4) Fuse Duplicates (Duplicate Fusion Subsystem) / (DFS)

After clicking the "Fuse Duplicates (DFS)" on the following screen, the system takes the user to the Fuse Duplicate Values (Duplicate Fusion Subsystem):

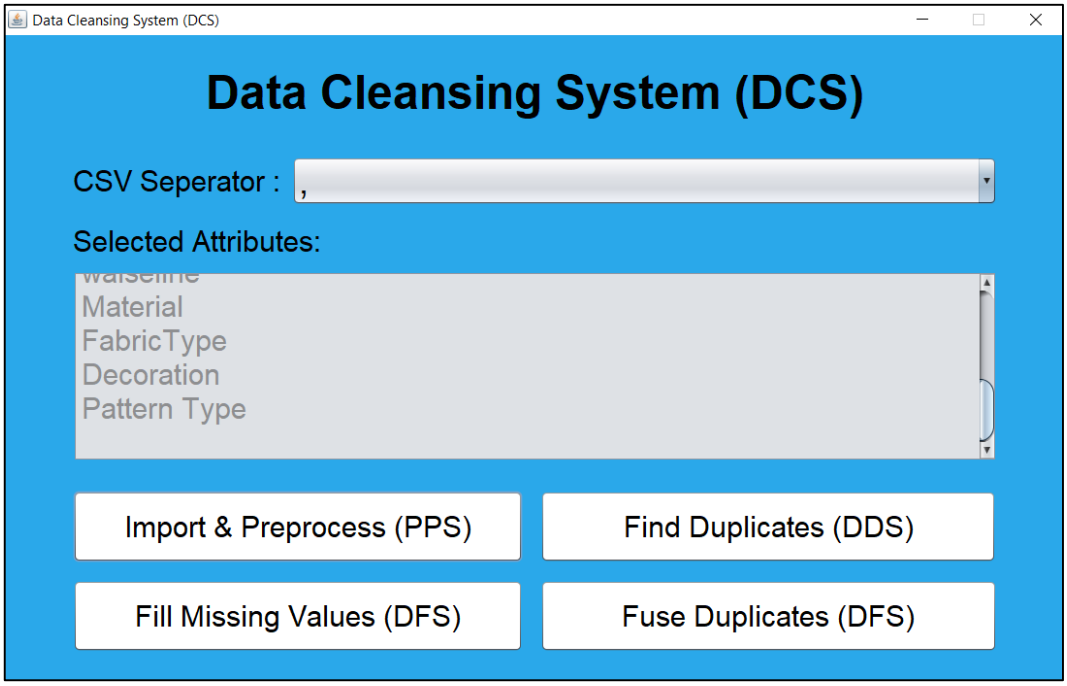

Fig. 39. Selected Attributes (DFS)

Fuse Duplicate Values (DFS) extracts duplicate pairs, analyzes them to merge them into a single record. During the data fusion, duplicate records are merged to create a new and more accurate and useful record as a result. Duplicate fusion technique not only resolves the unintended duplicates issue but also helps to merge the deliberately created duplicates from the parallel sensors setup. This technique is widely used for the fusion of multiple inputs received from the multiple sensors installed to record a similar event.

Duplicate Fusion Subsystem (DFS) enabled fusion options are based on the fusion operations calculated during the first iteration. After the selection of fusion options, all records are selected one by one and data cleansing is performed according to the values saved for each attribute of the selected record. Duplicate Fusion Subsystem (DFS) supports the **"Maximum Similarity Sum"** and **"Maximum Occurrences"** data fusion options. Runtime selection of one of these options is given to the user and the default option is executed on the system if nothing received from the user.

Fuse Duplicate Values interface given in Fig. 40 is shown for the user after clicking the "Fuse Duplicates (DFS)" button on the DCS screen is shown after preprocessing.

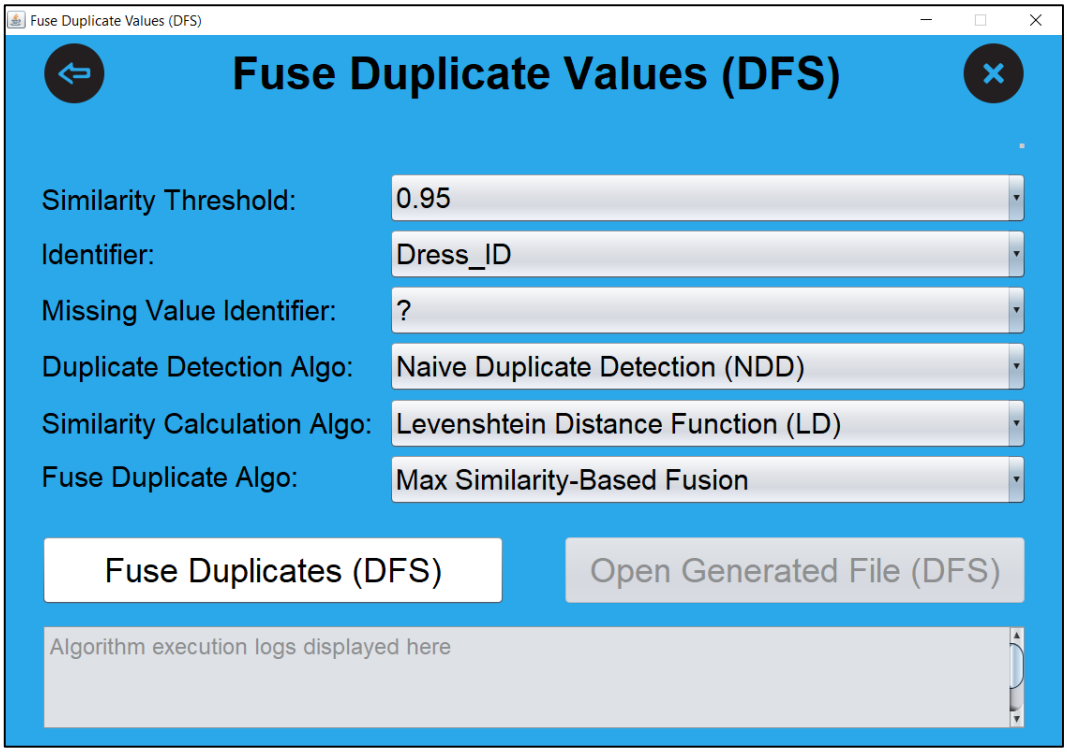

Fig. 40. Duplicate Fusion Subsystem (DFS)

Similarity threshold selection is customized for the Duplicate Fusion Subsystem (DFS) so that user can select the appropriate threshold from the options given in Fig. 41.

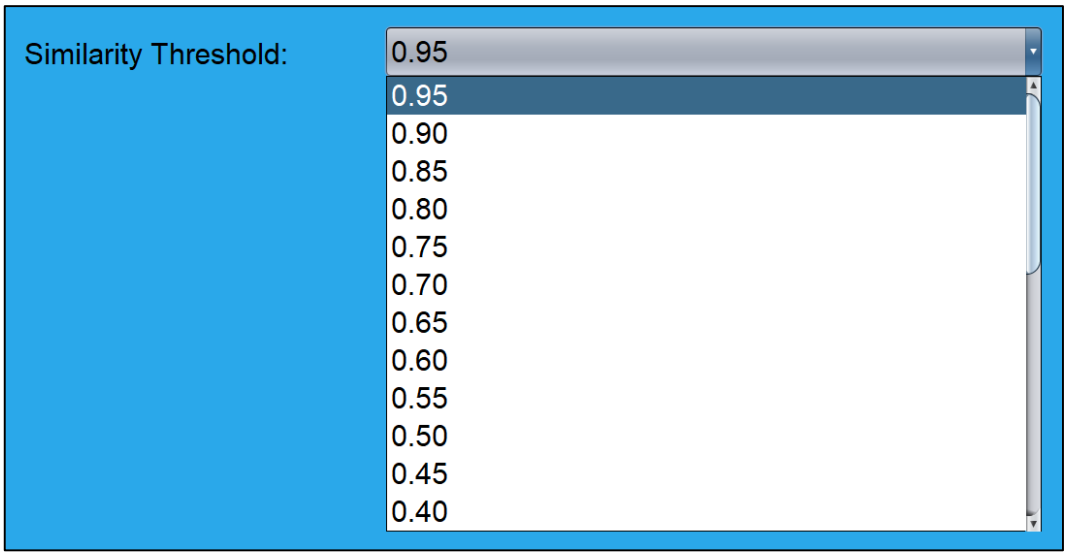

Fig. 41. Similarity Threshold Selection (DFS)

Dataset identifier is customized for the Duplicate Fusion Subsystem (DFS) so that user can select the appropriate identifier from the list of attributes as shown in Fig. 42. These attributes are extracted from the input dataset during the preprocessing step.

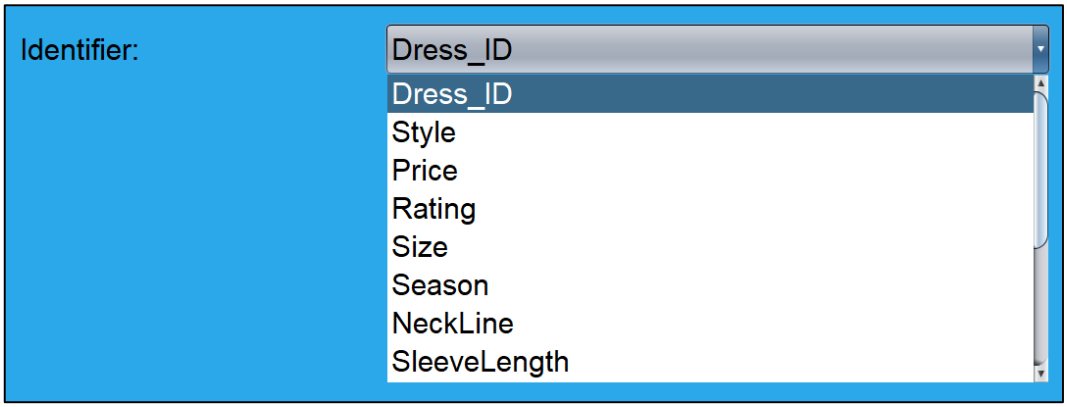

Fig. 42. Dataset Identifier Selection (DFS)

Missing value representation selection is customized for the Duplicate Fusion Subsystem (DFS) so that user can select the "?" or "null" as the missing value representation according to the input data as shown in Fig. 43.

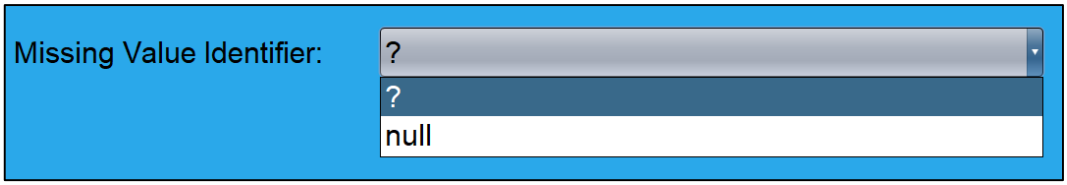

Fig. 43. Missing Value Representation Selection (DFS)

Duplicate detection algorithm selection is customized for the Duplicate Fusion Subsystem (DFS) so that user can select the appropriate duplicate detection algorithm from the options given in Fig. 44.

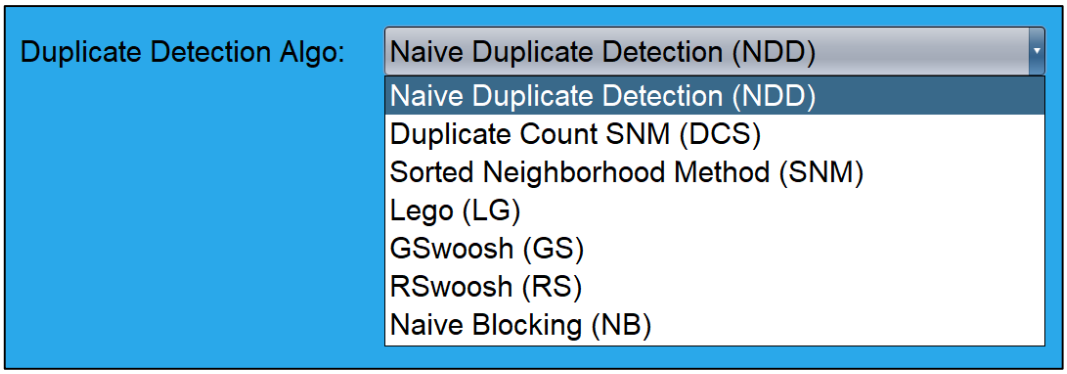

Fig. 44. Duplicate Detection Algorithm Selection (DFS)

Similarity calculation algorithm selection is customized for the Duplicate Fusion Subsystem (DFS) so that user can select the appropriate similarity calculation algorithm from the options given in Fig. 45.

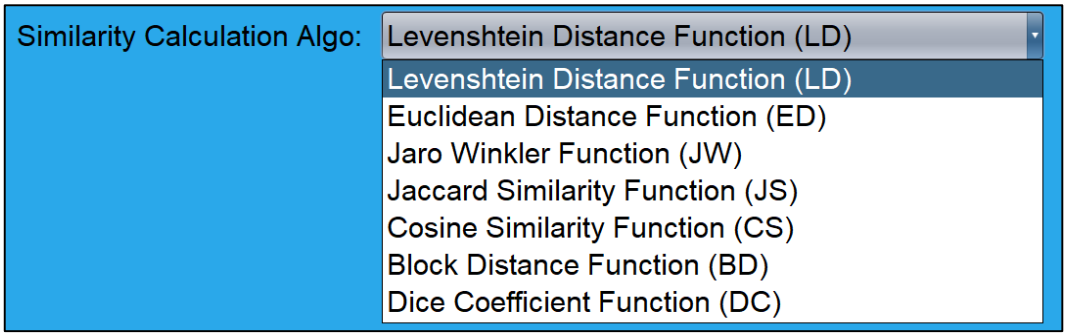

Fig. 45. Similarity Calculation Algorithm Selection (DFS)

Duplicate fusion algorithm selection is customized for the Duplicate Fusion Subsystem (DFS) so that user can select the suitable algorithm from the options given in Fig. 46.

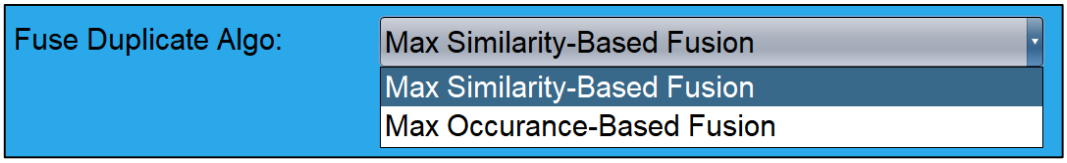

Fig. 46. Duplicate Fusion Algorithm Selection (DFS)

After clicking the "Fuse Duplicates (DFS)" button, the subsystem starts fusing the missing values of the input dataset as shown in Fig. 47.

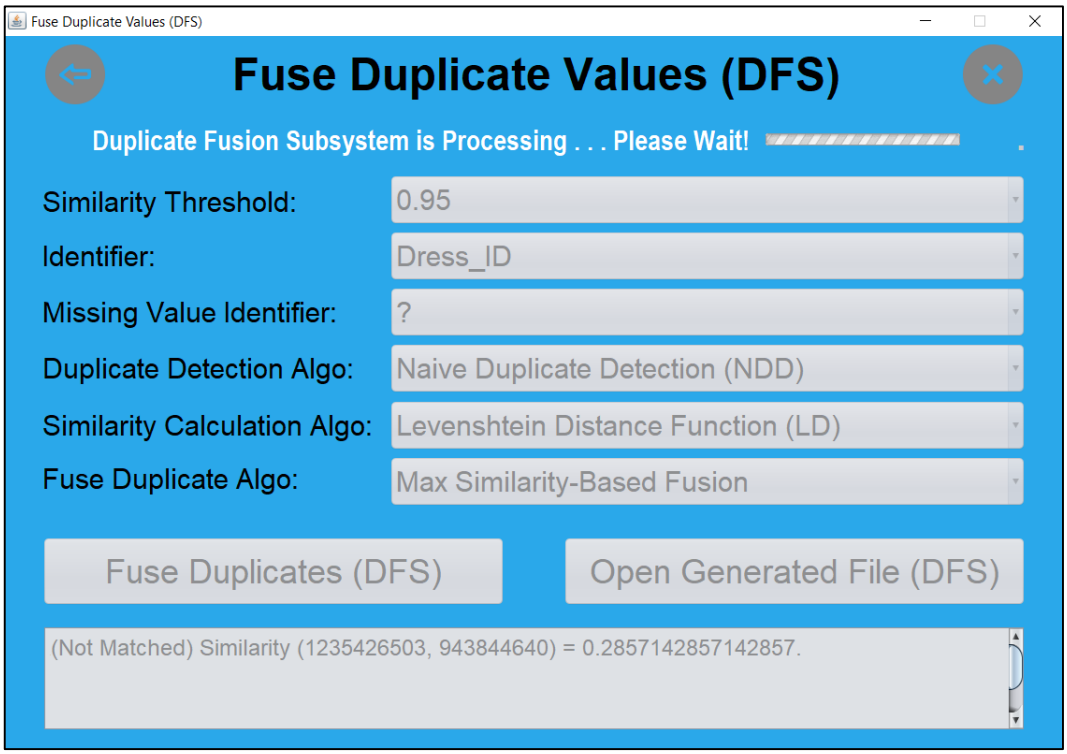

Fig. 47. DFS Processing

If the user clicks the close button during the processing of DFS, the message is displayed asking the user to wait for the processing as shown in Fig. 48.

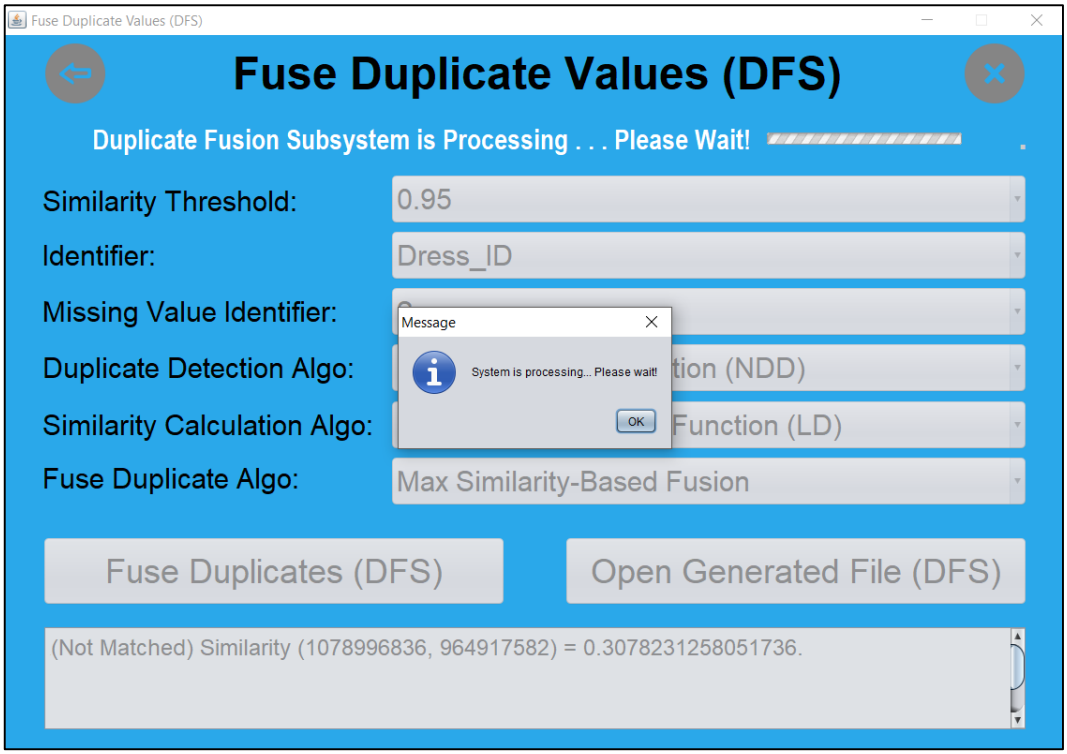

Fig. 48. Closing DFS Processing

A success message is displayed after fusing the duplicate values and writing the cleaned records of the input data set to the output file as shown in Fig. 49. “Open Generated File (DFS)” button is enabled to open the generated output file with the cleaned records.

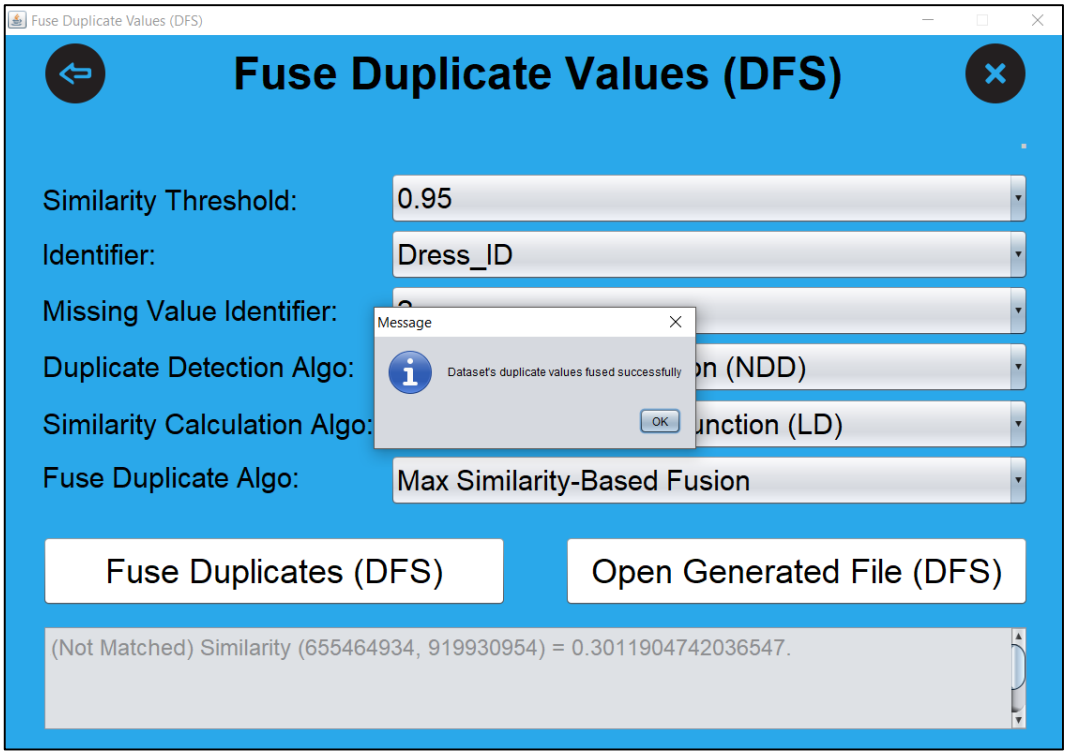

Fig. 49. DFS Processing Completed

Chapter 4

EXPERIMENTAL SETUP FOR VALIDATION

This chapter comprises the description of experiments setup used to get the results for the proposed algorithms in the research. The setup was created to validate the duplicate detection results, missing values filling results and duplicate fusion results for the correctness and endorsement of the contribution made by this research in the body of knowledge.

Data Cleansing System (DCS) is built on "Duplicate Detection toolkit (DuDe)" [7] and this toolkit contains the collection of duplicate detection algorithms written in Java programming language. DuDe was developed and tested by German data scientists over a period of time and continuous improvements were made to improve its quality. The quality and the open source license of this DuDe kit attracted us to get the benefits from it.

This work from this base point of "Duplicate Detection toolkit (DuDe)" [7] because it was already developed and tested for its performance and as per software engineering principle, the already done work was reused and reinvention of the wheel [92] was avoided. As per the "Quality in, quality out" [93] principle, the developed system is of high quality because the base and the input for the building block is of high quality.

4.1. Duplicate Detection (Validation)

Duplicate Detection Subsystem (DDS) integrates the "Duplicate Detection toolkit (DuDe)" [7] to find the duplicates from the given input data. Data Cleansing System (DCS) takes the data as input and preprocess it through the Pre-Processing Subsystem (PPS). DCS then gives this preprocessed data to DDS as input and duplicates are generated.

To verify the process of Duplicate Detection Subsystem, similar datasets were selected those were selected by the DuDe kit and following two algorithms were selected for the testing purpose:

1. "Naive Duplicate Detection" [7, 57]
2. "Levenshtein Distance Function" [7, 65]

4.2. Duplicate Fusion (Validation)

Duplicate fusion technique not only resolves the unintended duplicates issue but also helps to merge the deliberately created duplicates from the parallel sensors setup [49-52, 76, 77]. This technique is widely used for the fusion of multiple inputs received from the multiple sensors installed to record a similar event [50, 78].

Literature of data science tells that the improvements in the dataset quality, owing to effective data cleansing, can be measured with the classification accuracy. Classification accuracy is measured before and after the data cleansing process and increase in the classification accuracy states the effectiveness of the data cleansing process. Considering this verification technique, different data cleansing techniques are chosen to check their effectiveness with the help of classification accuracy improvements.

"Audiology" [94] and "Dresses Attribute Sales" (DASD) [95] datasets are chosen from the "UCI machine learning repository" [94] to test different data cleansing techniques and compare them with the proposed techniques. RapidMiner Studio [5], a well-known data mining tool, is selected to check the different data cleansing techniques. RapidMiner is developed in Java language and has extensive usage in the data science industry and scientific community. RapidMiner's Replace Missing Value Operator (RM-RMVO) [5] is selected to fill the missing values of the dataset.

Duplicate Fusion Subsystem (DFS) performs one of the below three operations on the input duplicate pairs to perform the data cleansing as per the given user input. Runtime selection of one of these options is given to the user and the default option is executed on the system if nothing received from the user. Duplicate Fusion Subsystem (DFS) supports the following three types of data fusion options:

- Merge & Fill (SimFiller, DuDeFiller)
- Maximum Similarity Sum (DuDeFuse)
- Maximum Occurrences (DuDeFuse)

SimFiller and DuDeFiller algorithms were selected from the developed algorithms to test the DCS for validation. Algorithm selection is customized and both DuDeFiller and DuDeFuse algorithms work in the DCS.

1) SimFiller

SimFiller is similarity-based missing values filling algorithm which runs independently on its own to fill the missing values of a dataset. This algorithm directly accepts the input dataset after Pre-Processing Subsystem (PPS), produces the pairs of similar records, filters them based on similarity threshold, check each pair to verify that at least one record's attribute contains a null value then fills the missing values within each pair by taking the relevant not null value.

SimFiller's technique is useful for data preprocessing to fix the inconsistencies in the dataset related to missing values. Quality of the dataset is increased after passing it from this data cleansing option and to verify this statement "Audiology" [94] dataset is chosen from the "UCI machine learning repository" [94]. This dataset comprises 26 testing and 200 training records and has 71 attributes including and class label and identifier.

Replicas of "Audiology" dataset are generated filled for the missing values using a specific value, zero value, maximum value, minimum value, the average value with the RapidMiner Studio [5] Replace Missing Value Operator (RMVO) and one replica is created after filing missing values with the SimFiller algorithm. Classification accuracy results of "Decision Tree", "Naive Bayes", "Deep Learning" and "Random Forest" are generated from each replica of "Audiology" dataset.

2) DuDeFiller / DuDeFuse

"Audiology" [94] and "Dresses Attribute Sales" (DASD) [95] datasets are selected from the "UCI machine learning repository" [94] to test the DuDeFiller/DuDeFuse algorithm. "Audiology" [94] dataset comprises 26 testing and 200 training records and have 71 attributes including and class label and identifier. "Dresses Attribute Sales" (DASD) [95] dataset contains 11 attributes excluding class label and identifier and 501 records.

RapidMiner Studio [5], a well-known data mining tool, is selected to check the different data cleansing techniques. RapidMiner is developed in Java language and has extensive usage in the data science industry and scientific community. RapidMiner's Replace Missing Value Operator (RM-RMVO) [5] options of specific value replacements, zero value replacements, maximum value replacements, minimum value replacements and the average value replacements are selected to fill the missing values of the dataset.

Replicas of “Audiology” [94] and “Dresses Attribute Sales” (DASD) [95] datasets are generated filled for the missing values using a specific value, zero value, maximum value, minimum value, the average value with the RapidMiner Studio [5] Replace Missing Value Operator (RMVO) and one replica is created after filing missing values with SimFiller algorithm. Classification accuracy results of “Decision Tree”, “Naive Bayes”, “Deep Learning” and “Random Forest” are generated from each replica of “Audiology” dataset.

10 folds’ cross validation based stratified sampling [96] is used to avoid the biases of dataset towards a specific class label in the classification results. Cross validation [96] splits the dataset into **x** number of divisions and considers **x-1** divisions for training and **1** division for testing in an iterative way until each division is used one time for testing. The accuracy of the dataset is logged during each iteration and complete accuracy is obtainable as average and as accuracy range from least to an extreme value.

Some data scientists state that the accuracy can never be the suitable measurement to verify the efficiency of any classifier because it overlooks the attention of false negative and false positive cases. These considerations false negative and false positive cases are made by the f-measure [97] which defines the correctness of classifier for a binomial class label dataset. “Dresses Attribute Sales dataset” (DASD) [95] has a binomial class label, therefore, f-measure of this dataset is also calculated to check the improvements in the classification correctness.

Chapter 5

RESULTS AND EVALUATION

Experiments are conducted after completing the evaluation setup and the achieved results are presented and discussed in this chapter. This chapter comprises the duplicate detection results, missing values filling results and duplicate fusion results of experiments carried out for the proposed system. Results show the correctness and validation of the contribution made by the research in the body of knowledge.

5.1. Results

Following four experimental results are discussed in this chapter for the algorithm proposed by this research in the previous chapters.

- SimFiller Classification Results for "Audiology" Dataset
    - Classification accuracy results of Decision Tree, "Naive Bayes", "Deep Learning" and "Random Forest" for the "Audiology" dataset filled with the SimFiller algorithm is presented in this section.
- DuDeFiller Classification Results for "Audiology" Dataset
    - Classification accuracy results of "Decision Tree", "Naive Bayes", "Deep Learning" and "Random Forest" for the "Audiology" dataset filled with the DuDeFiller algorithm is presented in this section.
- DuDeFiller Classification Results for "Dresses Attribute Sales Dataset" (DASD)
    - Classification accuracy results of "Decision Tree", "Naive Bayes", "Deep Learning", "Random Forest" and "Logistic Regression" for the "Dresses Attribute Sales dataset" (DASD) filled with the DuDeFiller algorithm is presented in this section.
- DuDeFiller F-measure Results for "Dresses Attribute Sales Dataset" (DASD)
    - Classification accuracy results of "Decision Tree", "Naive Bayes", "Deep Learning", "Random Forest" and "Logistic Regression" for the "Dresses Attribute Sales dataset" (DASD) filled with the DuDeFiller algorithm is presented in this section.

5.2. SimFiller Classification Results for Audiology Dataset

Classification accuracy results of following classifiers for the "Audiology" dataset filled with six different algorithms are presented in this section:

- Decision Tree
- Naive Bayes
- Deep Learning
- Random Forest

Six algorithms used for missing values replacements are Average, Minimum, Maximum, Zero, Specific Value and SimFiller.

1) Decision Tree

Classification accuracy of the "Decision Tree" classifier for "Audiology" dataset, filled for missing values with six different algorithms, is shown in Table 10 and Fig. 50. Classification accuracies of all six missing value replacement algorithms, on "Audiology" dataset, are 15.38% for Average as a replacement, 76.92% for Minimum as a replacement, 76.92% for Maximum as a replacement, 76.92% for Zero as a replacement, 15.38% for Specific Value as a replacement and 84.62% for replacement with the SimFiller algorithm.

	Classification Accuracy	**Classification Error**
Average	15.38%	84.62%
Minimum	76.92%	23.08%
Maximum	76.92%	23.08%
Zero	76.92%	23.08%
Value	15.38%	84.62%
SimFiller	84.62%	15.38%

Table 10. The accuracy of the "Decision Tree" for Audiology Dataset (SimFiller)

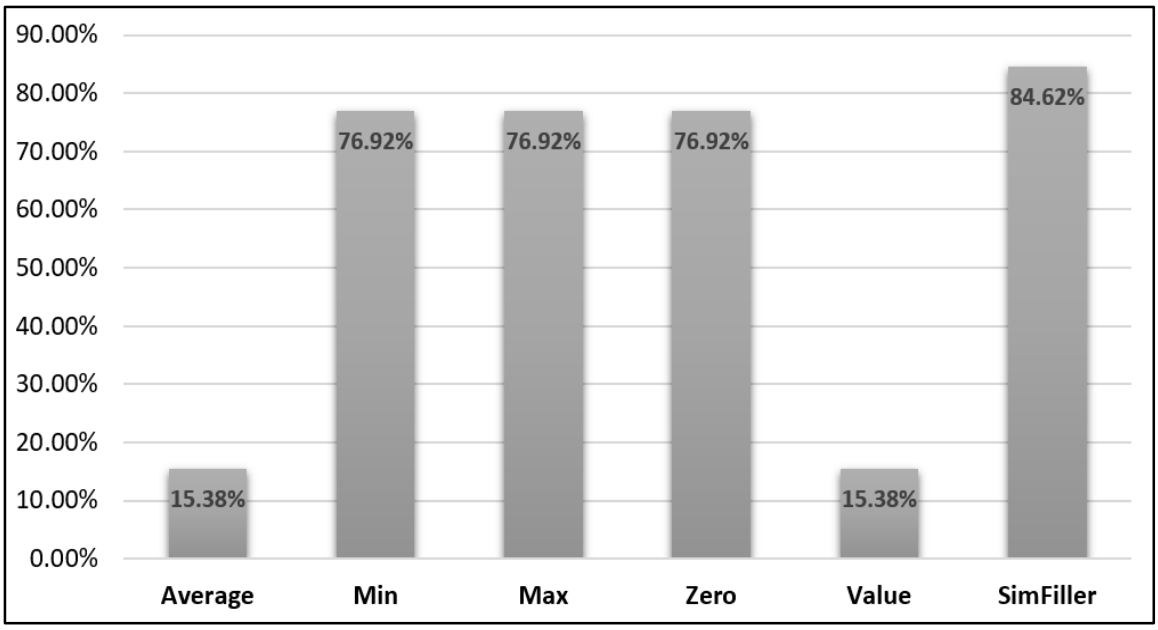

Fig. 50. The accuracy of the "Decision Tree" for Audiology Dataset (SimFiller)

2) Naive Bayes

Classification accuracy of the "Naive Bayes" classifier for "Audiology" dataset, filled for missing values with six different algorithms, is shown in Table 11 and Fig. 51. Classification accuracies of all six missing value replacement algorithms, on "Audiology" dataset, are 61.54% for Average as a replacement, 69.23% for Minimum as a replacement, 69.23% for Maximum as a replacement, 69.23% for Zero as a replacement, 65.38% for Specific Value as a replacement and 80.77% for replacement with the SimFiller algorithm.

	Classification Accuracy	**Classification Error**
Average	61.54%	38.46%
Minimum	69.23%	30.77%
Maximum	69.23%	30.77%
Zero	69.23%	30.77%
Value	65.38%	34.62%
SimFiller	80.77%	19.23%

Table 11. The accuracy of "Naive Bayes" for Audiology Dataset (SimFiller)

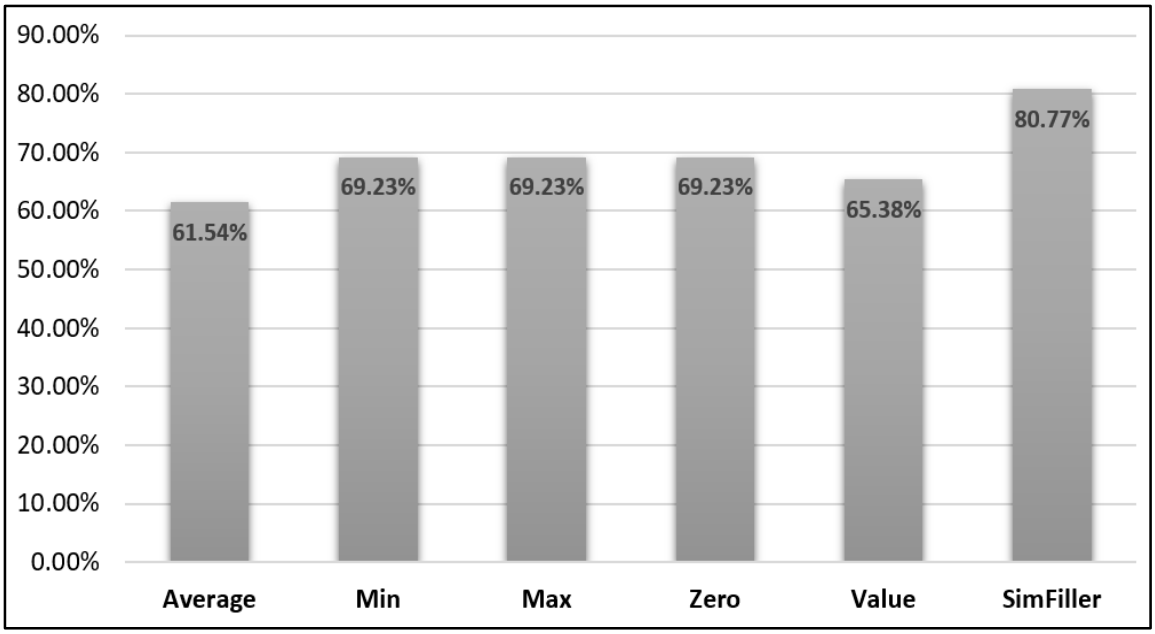

Fig. 51. The accuracy of "Naive Bayes" for Audiology Dataset (SimFiller)

3) Deep Learning

Classification accuracy of the "Deep Learning" classifier for "Audiology" dataset, filled for missing values with six different algorithms, is shown in Table 12 and Fig. 52. Classification accuracies of all six missing value replacement algorithms, on "Audiology" dataset, are 76.92% for Average as a replacement, 73.08% for Minimum as a replacement, 73.08% for Maximum as a replacement, 73.08% for Zero as a replacement, 61.54% for Specific Value as a replacement and 88.46% for replacement with the SimFiller algorithm.

	Classification Accuracy	**Classification Error**
Average	76.92%	23.08%
Minimum	73.08%	26.92%
Maximum	73.08%	26.92%
Zero	73.08%	26.92%
Value	61.54%	38.46%
SimFiller	88.46%	11.54%

Table 12. The accuracy of "Deep Learning" for Audiology Dataset (SimFiller)

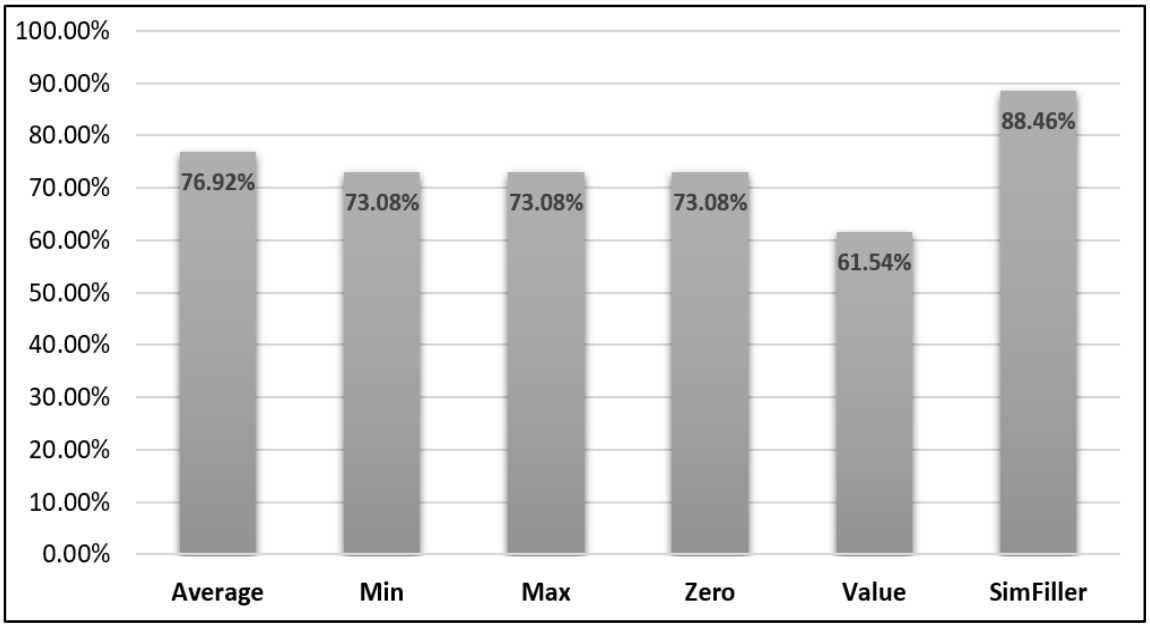

Fig. 52. The accuracy of "Deep Learning" for Audiology Dataset (SimFiller)

4) Random Forest

Classification accuracy of the "Random Forest" classifier for "Audiology" dataset, filled for missing values with six different algorithms, is shown in Table 13 and Fig. 53. Classification accuracies of all six missing value replacement algorithms, on "Audiology" dataset, are 53.85% for Average as a replacement, 50.00% for Minimum as a replacement, 50.00% for Maximum as a replacement, 50.00% for Zero as a replacement, 50.00% for Specific Value as a replacement and 61.54% for replacement with the SimFiller algorithm.

	Classification Accuracy	**Classification Error**
Average	53.85%	46.15%
Minimum	50.00%	50.00%
Maximum	50.00%	50.00%
Zero	50.00%	50.00%
Value	50.00%	50.00%
SimFiller	61.54%	38.46%

Table 13. The accuracy of "Random Forest" for Audiology Dataset (SimFiller)

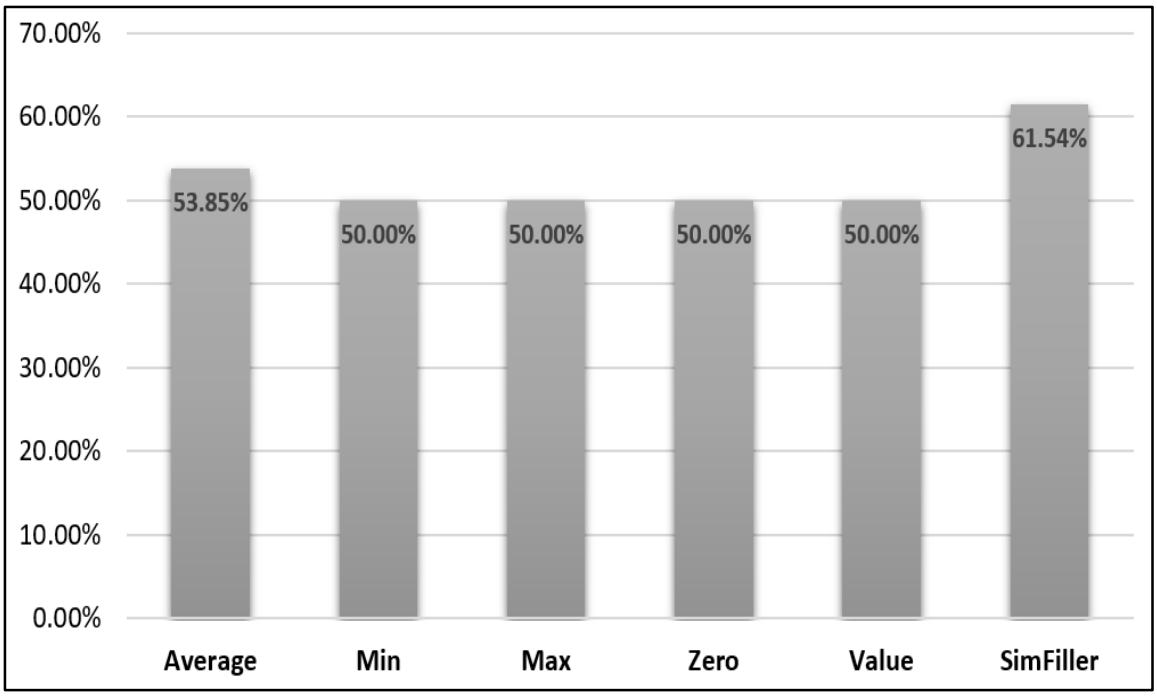

Fig. 53. The accuracy of "Random Forest" for Audiology Dataset (SimFiller)

5.3. DuDeFiller Classification Results for Audiology Dataset

To test the DuDeFiller for the Audiology dataset, the classification accuracy of the following classifiers for the "Audiology" dataset is calculated. Multiple copies of Audiology dataset are created those were filled with six different missing values filling algorithms:

- Decision Tree
- Naive Bayes
- Deep Learning
- Random Forest

Six algorithms used for missing values replacements are:

- Average
- Minimum
- Maximum
- Zero
- Specific Value
- DuDeFiller

1) Decision Tree

Classification accuracy of the "Decision Tree" classifier for "Audiology" dataset, filled for missing values with six different algorithms, is shown in Table 14 and Fig. 54. Classification accuracies of all six missing value replacement algorithms, on "Audiology" dataset, are 67.26% for Average as a replacement, 68.14% for Minimum as a replacement, 68.14% for Maximum as a replacement, 68.14% for Zero as a replacement, 68.14% for Specific Value as a replacement and 69.03% for replacement with the DuDeFiller algorithm.

	Classification Accuracy	**Classification Error**
Average	67.41% +/- 21.99% (micro average: 67.26%)	32.59% +/- 21.99% (micro average: 32.74%)
Minimum	68.28% +/- 22.23% (micro average: 68.14%)	31.72% +/- 22.23% (micro average: 31.86%)
Maximum	68.28% +/- 22.23% (micro average: 68.14%)	31.72% +/- 22.23% (micro average: 31.86%)
Zero	68.28% +/- 22.23% (micro average: 68.14%)	31.72% +/- 22.23% (micro average: 31.86%)
Value	68.28% +/- 22.23% (micro average: 68.14%)	31.72% +/- 22.23% (micro average: 31.86%)
DuDeFiller	**69.17% +/- 22.65%** **(micro average: 69.03%)**	**30.83% +/- 22.65%** **(micro average: 30.97%)**

Table 14. The accuracy of the "Decision Tree" for Audiology Dataset (DuDeFiller)

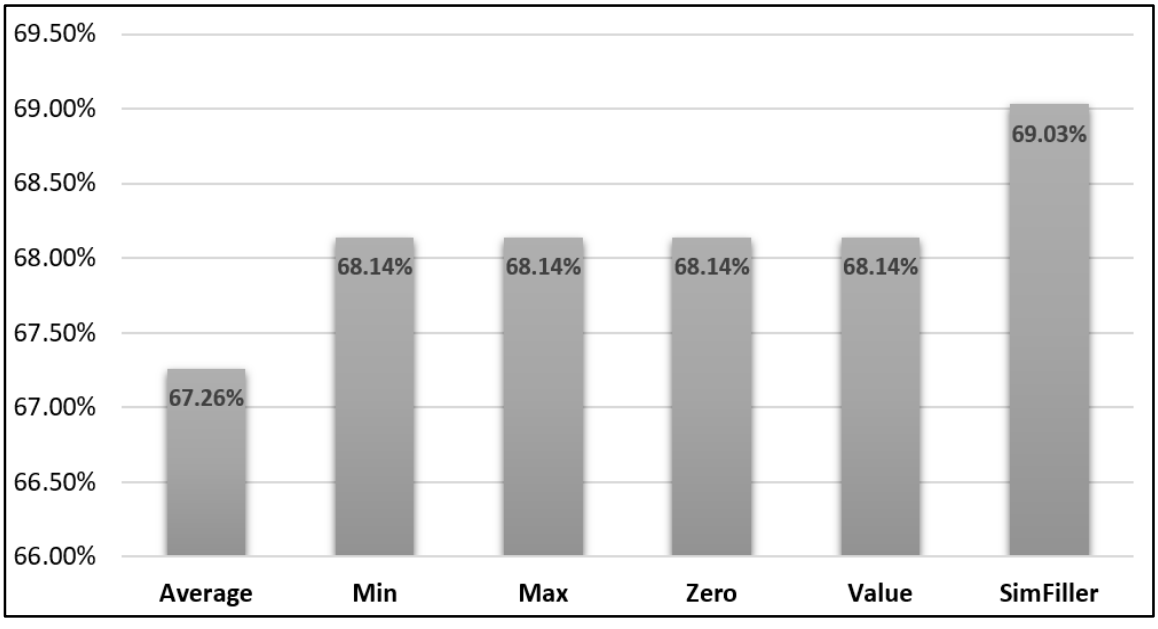

Fig. 54. The accuracy of the "Decision Tree" for Audiology Dataset (DuDeFiller)

2) Naive Bayes

Classification accuracy of the "Naive Bayes" classifier for "Audiology" dataset, filled for missing values with six different algorithms, is shown in Table 15 and Fig. 55. Classification accuracies of all six missing value replacement algorithms, on "Audiology" dataset, are 80.97% for Average as a replacement, 80.97% for Minimum as a replacement, 80.97% for Maximum as a replacement, 80.97% for Zero as a replacement, 80.53% for Specific Value as a replacement and 80.97% for replacement with the DuDeFiller algorithm.

	Classification Accuracy	**Classification Error**
Average	80.93% +/- 6.68% (micro average: 80.97%)	19.07% +/- 6.68% (micro average: 19.03%)
Minimum	80.95% +/- 6.36% (micro average: 80.97%)	19.05% +/- 6.08% (micro average: 19.03%)
Maximum	80.95% +/- 6.36% (micro average: 80.97%)	19.05% +/- 6.08% (micro average: 19.03%)
Zero	80.95% +/- 6.36% (micro average: 80.97%)	19.05% +/- 6.08% (micro average: 19.03%)
Value	80.49% +/- 6.11% (micro average: 80.53%)	19.51% +/- 6.11% (micro average: 19.47%)
DuDeFiller	**81.01% +/- 5.78% (micro average: 80.97%)**	**18.99% +/- 5.78% (micro average: 19.03%)**

Table 15. The accuracy of "Naive Bayes" for Audiology Dataset (DuDeFiller)

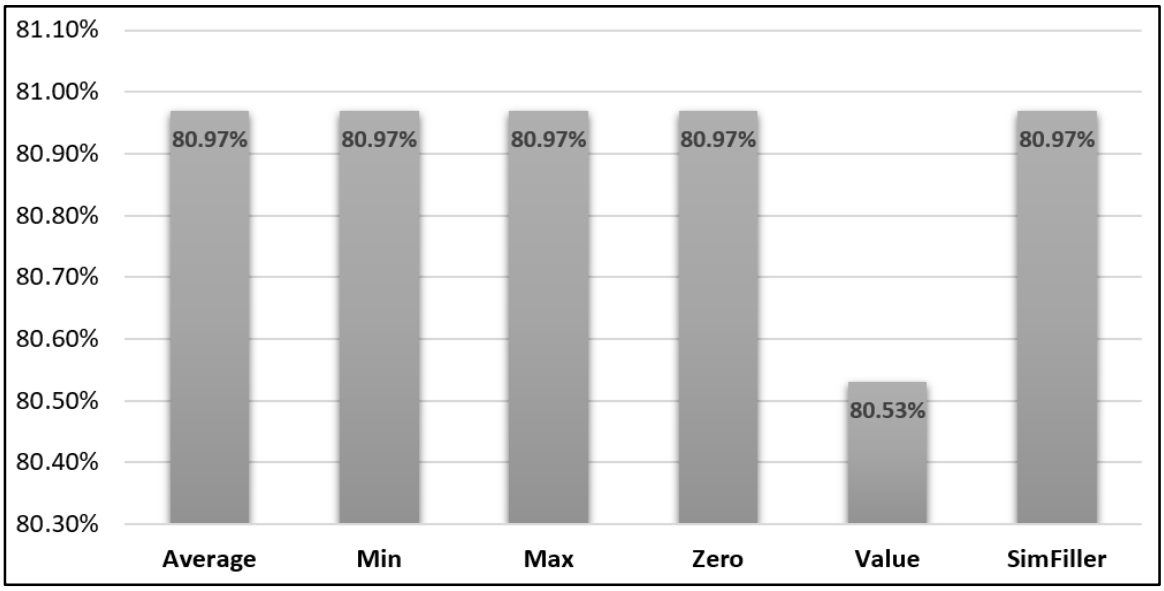

Fig. 55. The accuracy of "Naive Bayes" for Audiology Dataset (DuDeFiller)

3) Deep Learning

Classification accuracy of the "Deep Learning" classifier for "Audiology" dataset, filled for missing values with six different algorithms, is shown in Table 16 and Fig. 56. Classification accuracies of all six missing value replacement algorithms, on "Audiology" dataset, are 74.34% for Average as a replacement, 70.80% for Minimum as a replacement, 70.80% for Maximum as a replacement, 70.80% for Zero as a replacement, 73.45% for Specific Value as a replacement and 74.78% for replacement with the DuDeFiller algorithm.

	Classification Accuracy	**Classification Error**
Average	74.27% +/- 7.59% (micro average: 74.34%)	25.73% +/- 7.59% (micro average: 25.66%)
Minimum	70.75% +/- 7.04% (micro average: 70.80%)	29.25% +/- 7.04% (micro average: 29.20%)
Maximum	70.75% +/- 7.04% (micro average: 70.80%)	29.25% +/- 7.04% (micro average: 29.20%)
Zero	70.75% +/- 7.04% (micro average: 70.80%)	29.25% +/- 7.04% (micro average: 29.20%)
Value	73.42% +/- 6.93% (micro average: 73.45%)	26.58% +/- 6.93% (micro average: 26.55%)
DuDeFiller	**74.76% +/- 4.48% (micro average: 74.78%)**	**25.24% +/- 4.48% (micro average: 25.22%)**

Table 16. The accuracy of "Deep Learning" for Audiology Dataset (DuDeFiller)

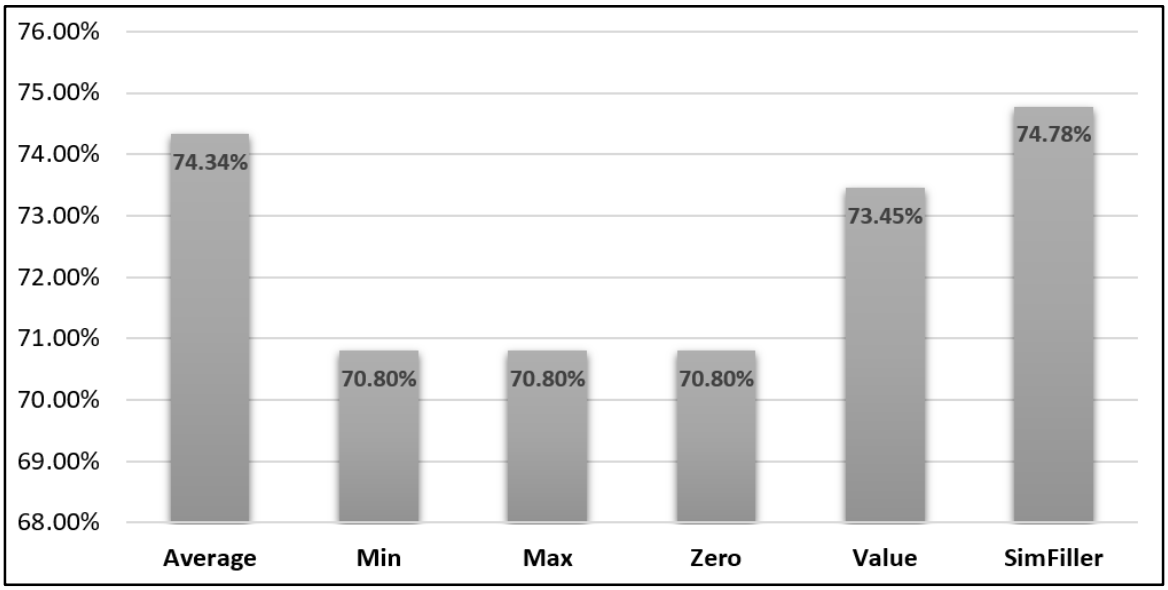

Fig. 56. The accuracy of "Deep Learning" for Audiology Dataset (DuDeFiller)

4) Random Forest

Classification accuracy of the "Random Forest" classifier for "Audiology" dataset, filled for missing values with six different algorithms, is shown in Table 17 and Fig. 57. Classification accuracies of all six missing value replacement algorithms, on "Audiology" dataset, are 71.68% for Average as a replacement, 73.01% for Minimum as a replacement, 73.01% for Maximum as a replacement, 73.01% for Zero as a replacement, 73.01% for Specific Value as a replacement and 75.22% for replacement with the DuDeFiller algorithm.

	Classification Accuracy	**Classification Error**
Average	71.66% +/- 5.72% (micro average: 71.68%)	28.34% +/- 5.72% (micro average: 28.32%)
Minimum	72.98% +/- 6.24% (micro average: 73.01%)	27.02% +/- 6.24% (micro average: 26.99%)
Maximum	72.98% +/- 6.24% (micro average: 73.01%)	27.02% +/- 6.24% (micro average: 26.99%)
Zero	72.98% +/- 6.24% (micro average: 73.01%)	27.02% +/- 6.24% (micro average: 26.99%)
Value	72.98% +/- 6.24% (micro average: 73.01%)	27.02% +/- 6.24% (micro average: 26.99%)
DuDeFiller	**75.26% +/- 4.31% (micro average: 75.22%)**	**24.74% +/- 4.31% (micro average: 24.78%)**

Table 17. The accuracy of "Random Forest" for Audiology Dataset (DuDeFiller)

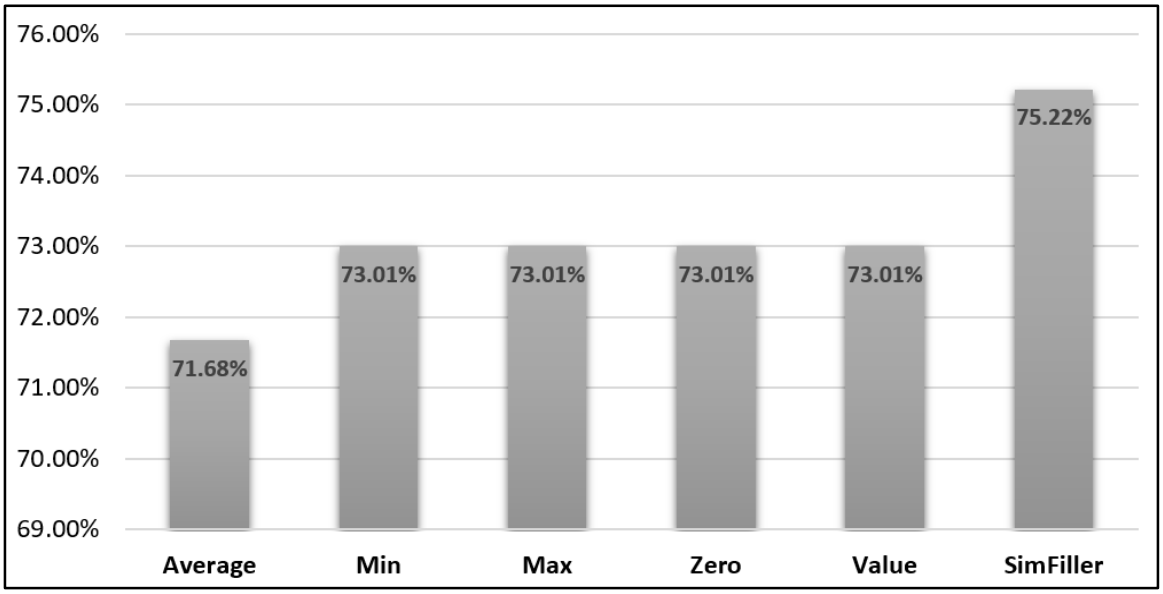

Fig. 57. The accuracy of "Random Forest" for Audiology Dataset (DuDeFiller)

5.4. DuDeFiller Classification Results for Dresses Attribute Sales Dataset (DASD)

DuDeFiller is tested for the classification accuracy results of following classifiers for the "Dresses Attribute Sales dataset" (DASD). Six copies of DASD filled with six different algorithms are presented in this section:

- Decision Tree
- Naive Bayes
- Deep Learning
- Random Forest
- Logistic Regression

Six algorithms used for creating the six copies of "Dresses Attribute Sales Dataset" (DASD) with missing values replacements are:

- Average
    - DASD missing values are filled with the average value replacements after calculating the average value for each attribute's missing value.
- Minimum
    - DASD missing values are filled with the minimum value replacements after calculating the minimum value for each attribute's missing value.
- Maximum
    - A maximum value is calculated from the completed data and is placed for all the missing values of that attribute in DASD.
- Zero
    - Zero is placed for all the missing values of the DASD.
- Specific Value
    - Specific Value is defined for the DASD and is placed for all the missing values.
- DuDeFiller
    - The developed duplicate detection based missing value filling algorithm calculates the missing value of DASD.

1) Decision Tree

Classification accuracy of the "Decision Tree" classifier for "Dresses Attribute Sales Dataset" (DASD), filled for missing values with six different algorithms, is shown in Table 18 and Fig. 58. Classification accuracies of all six missing value replacement algorithms, on DASD, are 58.00% for Average as a replacement, 58.00% for Minimum as a replacement, 58.00% for Maximum as a replacement, 58.00% for Zero as a replacement, 58.00% for Specific Value as a replacement and 58.20% for replacement with the DuDeFiller algorithm.

	Classification Accuracy	**Classification Error**
Average	58.00% +/- 0.00% (micro average: 58.00%)	42.00% +/- 0.00% (micro average: 42.00%)
Minimum	58.00% +/- 0.00% (micro average: 58.00%)	42.00% +/- 0.00% (micro average: 42.00%)
Maximum	58.00% +/- 0.00% (micro average: 58.00%)	42.00% +/- 0.00% (micro average: 42.00%)
Zero	58.00% +/- 0.00% (micro average: 58.00%)	42.00% +/- 0.00% (micro average: 42.00%)
Value	58.00% +/- 0.00% (micro average: 58.00%)	42.00% +/- 0.00% (micro average: 42.00%)
DuDeFiller	**58.20% +/- 0.6%** **(micro average: 58.20%)**	**41.80% +/- 0.60%** **(micro average: 41.80%)**

Table 18. The accuracy of the "Decision Tree" for DASD (DuDeFiller)

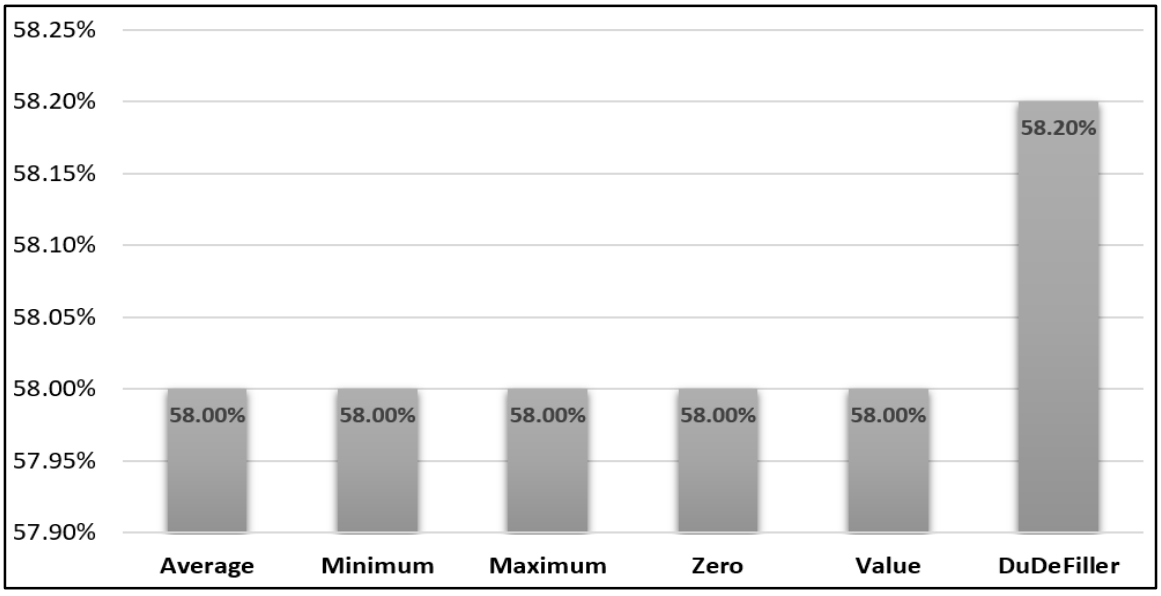

Fig. 58. The accuracy of the "Decision Tree" for DASD (DuDeFiller)

2) Naive Bayes

Classification accuracy of the "Naive Bayes" classifier for "Dresses Attribute Sales dataset" (DASD), filled for missing values with six different algorithms, is shown in Table 19 and Fig. 59. Classification accuracies of all six missing value replacement algorithms, on DASD, are 58.20% for Average as a replacement, 58.20% for Minimum as a replacement, 58.20% for Maximum as a replacement, 58.20% for Zero as a replacement, 58.20% for Specific Value as a replacement and 61.20% for replacement with the DuDeFiller algorithm.

	Classification Accuracy	**Classification Error**
Average	58.20% +/- 8.92% (micro average: 58.20%)	41.80% +/- 8.92% (micro average: 41.80%)
Minimum	58.20% +/- 8.92% (micro average: 58.20%)	41.80% +/- 8.92% (micro average: 41.80%)
Maximum	58.20% +/- 8.92% (micro average: 58.20%)	41.80% +/- 8.92% (micro average: 41.80%)
Zero	58.20% +/- 8.92% (micro average: 58.20%)	41.80% +/- 8.92% (micro average: 41.80%)
Value	58.20% +/- 8.92% (micro average: 58.20%)	41.80% +/- 8.92% (micro average: 41.80%)
DuDeFiller	**61.20% +/- 7.17% (micro average: 61.20%)**	**38.80% +/- 7.17% (micro average: 38.80%)**

Table 19. The accuracy of "Naive Bayes" for DASD (DuDeFiller)

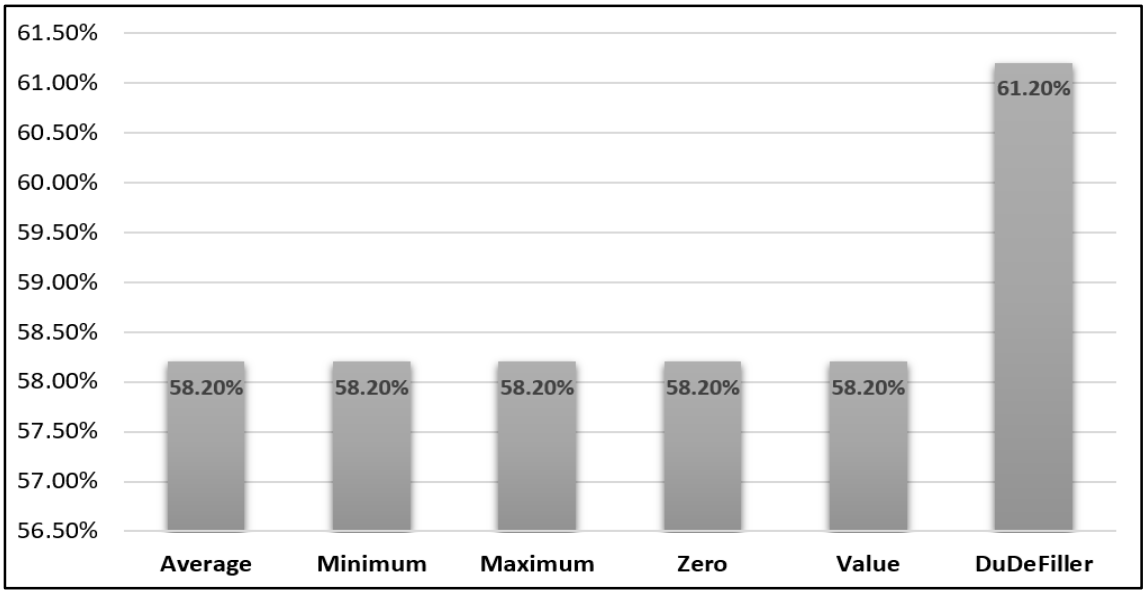

Fig. 59. The accuracy of "Naive Bayes" for DASD (DuDeFiller)

3) Deep Learning

Classification accuracy of the "Deep Learning" classifier for "Dresses Attribute Sales dataset" (DASD), filled for missing values with six different algorithms, is shown in Table 20 and Fig. 60. Classification accuracies of all six missing value replacement algorithms, on DASD, are 60.20% for Average as a replacement, 60.20% for Minimum as a replacement, 60.20% for Maximum as a replacement, 60.20% for Zero as a replacement, 61.40% for Specific Value as a replacement and 63.80% for replacement with the DuDeFiller algorithm.

	Classification Accuracy	**Classification Error**
Average	60.20% +/- 6.66% (micro average: 60.20%)	39.80% +/- 6.66% (micro average: 39.80%)
Minimum	60.20% +/- 6.66% (micro average: 60.20%)	39.80% +/- 6.66% (micro average: 39.80%)
Maximum	60.20% +/- 6.66% (micro average: 60.20%)	39.80% +/- 6.66% (micro average: 39.80%)
Zero	60.20% +/- 6.66% (micro average: 60.20%)	39.80% +/- 6.66% (micro average: 39.80%)
Value	61.40% +/- 8.49% (micro average: 61.40%)	38.60% +/- 8.49% (micro average: 38.60%)
DuDeFiller	**63.80% +/- 5.83% (micro average: 63.80%)**	**36.20% +/- 5.83% (micro average: 36.20%)**

Table 20. The accuracy of "Deep Learning" for DASD (DuDeFiller)

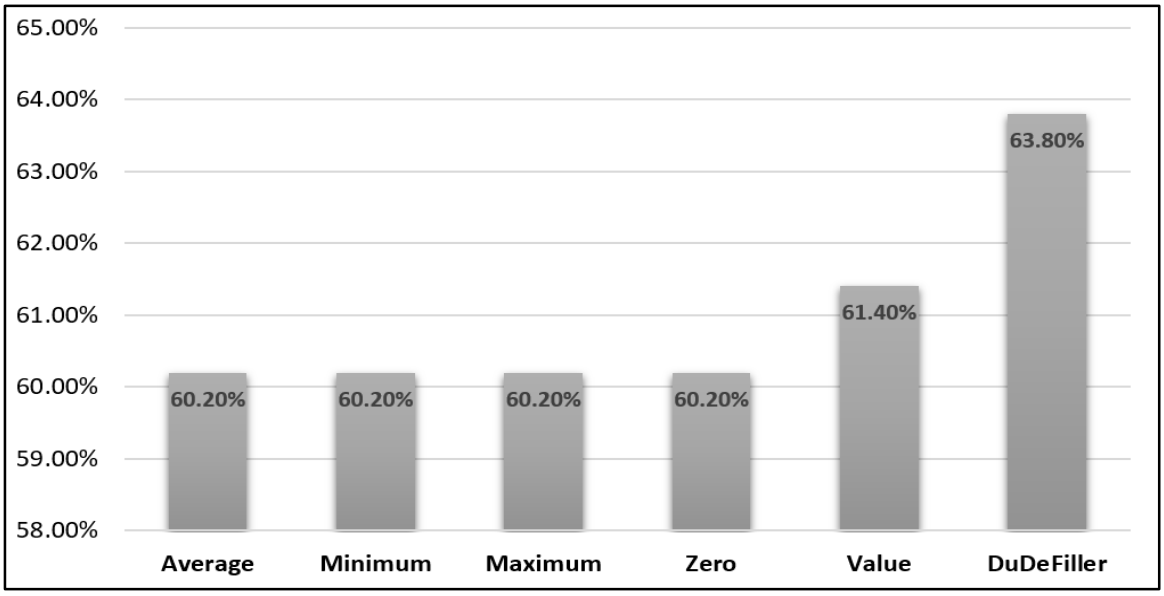

Fig. 60. The accuracy of "Deep Learning" for DASD (DuDeFiller)

4) Random Forest

Classification accuracy of the "Random Forest" classifier for "Dresses Attribute Sales dataset" (DASD), filled for missing values with six different algorithms, is shown in Table 21 and Fig. 61. Classification accuracies of all six missing value replacement algorithms, on DASD, are 58.00% for Average as a replacement, 58.00% for Minimum as a replacement, 58.00% for Maximum as a replacement, 58.00% for Zero as a replacement, 58.00% for Specific Value as a replacement and 58.20% for replacement with the DuDeFiller algorithm.

	Classification Accuracy	**Classification Error**
Average	58.00% +/- 0.0% (micro average: 58.00%)	42.00% +/- 0.00% (micro average: 42.00%)
Minimum	58.00% +/- 0.0% (micro average: 58.00%)	42.00% +/- 0.00% (micro average: 42.00%)
Maximum	58.00% +/- 0.0% (micro average: 58.00%)	42.00% +/- 0.00% (micro average: 42.00%)
Zero	58.00% +/- 0.0% (micro average: 58.00%)	42.00% +/- 0.00% (micro average: 42.00%)
Value	58.00% +/- 0.0% (micro average: 58.00%)	42.00% +/- 0.00% (micro average: 42.00%)
DuDeFiller	**58.20% +/- 0.6% (micro average: 58.20%)**	**41.80% +/- 0.60% (micro average: 41.80%)**

Table 21. The accuracy of "Random Forest" for DASD (DuDeFiller)

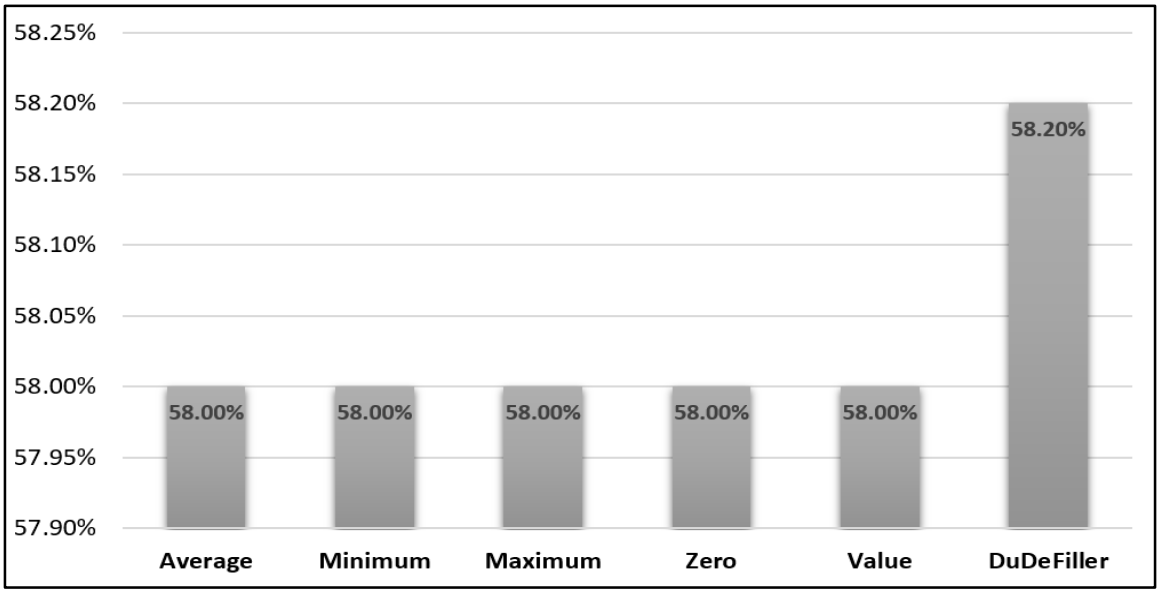

Fig. 61. The accuracy of "Random Forest" for DASD (DuDeFiller)

5) Logistic Regression

Classification accuracy of the "Logistic Regression" classifier for "Dresses Attribute Sales dataset" (DASD), filled for missing values with six different algorithms, is shown in Table 22 and Fig. 62. Classification accuracies of all six missing value replacement algorithms, on DASD, are 56.40% for Average as a replacement, 56.40% for Minimum as a replacement, 56.40% for Maximum as a replacement, 56.40% for Zero as a replacement, 56.40% for Specific Value as a replacement and 57.60% for replacement with the DuDeFiller algorithm.

	Classification Accuracy	**Classification Error**
Average	56.40% +/- 7.26% (micro average: 56.40%)	43.60% +/- 7.26% (micro average: 43.60%)
Minimum	56.40% +/- 7.26% (micro average: 56.40%)	43.60% +/- 7.26% (micro average: 43.60%)
Maximum	56.40% +/- 7.26% (micro average: 56.40%)	43.60% +/- 7.26% (micro average: 43.60%)
Zero	56.40% +/- 7.26% (micro average: 56.40%)	43.60% +/- 7.26% (micro average: 43.60%)
Value	56.40% +/- 7.26% (micro average: 56.40%)	43.60% +/- 7.26% (micro average: 43.60%)
DuDeFiller	**57.60% +/- 7.14% (micro average: 57.60%)**	**42.40% +/- 7.14% (micro average: 42.40%)**

Table 22. The accuracy of "Random Forest" for DASD (DuDeFiller)

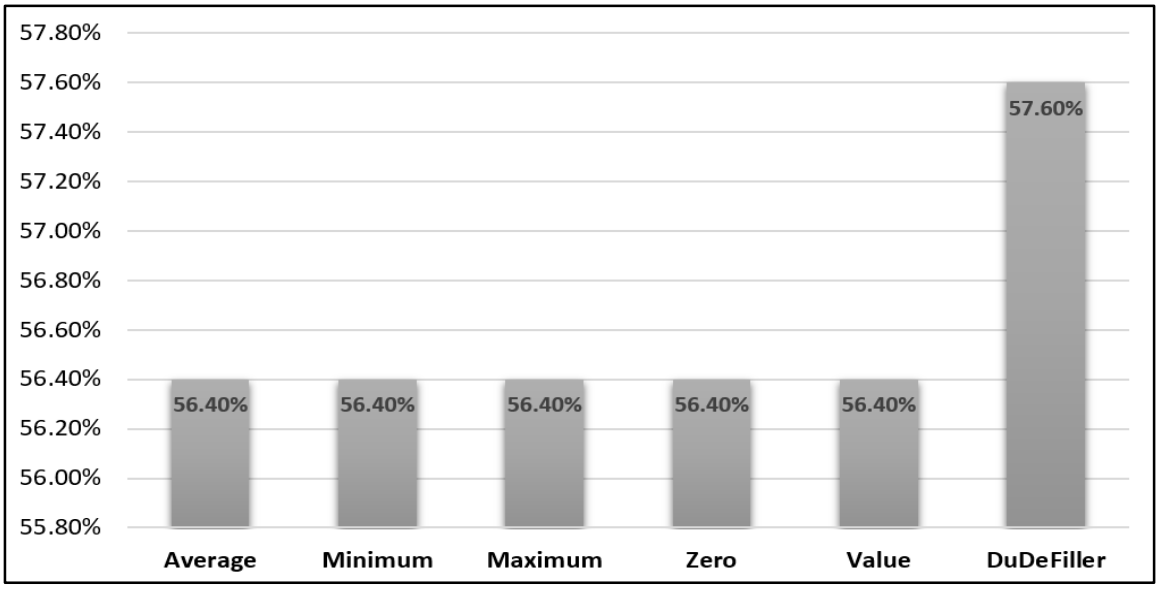

Fig. 62. The accuracy of "Random Forest" for DASD (DuDeFiller)

5.5. DuDeFiller F-measure Results for Dresses Attribute Sales Dataset (DASD)

Six algorithms used for missing values replacements are:

- Average
    - DASD missing values are filled with the average value replacements after calculating the average value for each attribute's missing value.
- Minimum
    - DASD missing values are filled with the minimum value replacements after calculating the minimum value for each attribute's missing value.
- Maximum
    - A maximum value is calculated from the completed data and is placed for all the missing values of that attribute in DASD.
- Zero
    - Zero is placed for all the missing values of the DASD.
- Specific Value
    - Specific Value is defined for the DASD and is placed for all the missing values.
- DuDeFiller
    - The developed duplicate detection based missing value filling algorithm calculates the missing value of DASD.

Six copies of "Dresses Attribute Sales Dataset" (DASD) are created by filling its missing values with the six algorithms shown above. F-measure results of following classifiers for the DASD is presented in this section:

- Decision Tree
- Naive Bayes
- Deep Learning
- Random Forest
- Logistic Regression

1) Decision Tree

The f-measure of the "Decision Tree" classifier for "Dresses Attribute Sales Dataset" (DASD), filled for missing values with six different algorithms, is shown in Table 23 and Fig. 63. The f-measure of all six missing value replacement algorithms, on DASD, are 73.42% for Average as a replacement, 73.42% for Minimum as a replacement, 73.42% for Maximum as a replacement, 73.42% for Zero as a replacement, 73.42% for Specific Value as a replacement and 73.58% for replacement with the DuDeFiller algorithm.

	F-measure
Average	73.42% +/- 0.0% (micro average: 73.42%) (positive class: 0)
Minimum	73.42% +/- 0.0% (micro average: 73.42%) (positive class: 0)
Maximum	73.42% +/- 0.0% (micro average: 73.42%) (positive class: 0)
Zero	73.42% +/- 0.0% (micro average: 73.42%) (positive class: 0)
Value	73.42% +/- 0.0% (micro average: 73.42%) (positive class: 0)
DuDeFiller	**73.58% +/- 0.47%** **(micro average: 73.58%) (positive class: 0)**

Table 23. The f-measure of "Decision Tree" for DASD (DuDeFiller)

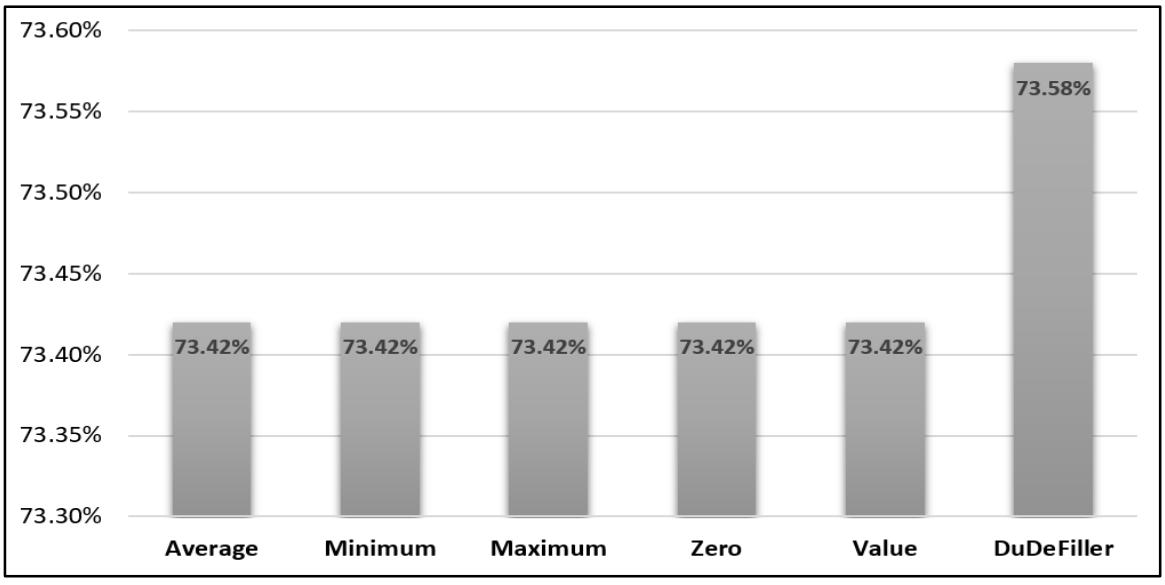

Fig. 63. The f-measure of "Decision Tree" for DASD (DuDeFiller)

2) Naive Bayes

The f-measure of the "Naive Bayes" classifier for "Dresses Attribute Sales Dataset" (DASD), filled for missing values with six different algorithms, is shown in Table 24 and Fig. 64. The f-measure of all six missing value replacement algorithms, on DASD, are 66.24% for Average as a replacement, 66.24% for Minimum as a replacement, 66.24% for Maximum as a replacement, 66.24% for Zero as a replacement, 66.24% for Specific Value as a replacement and 69.50% for replacement with the DuDeFiller algorithm.

	F-measure
Average	65.79% +/- 8.95% (micro average: 66.24%) (positive class: 0)
Minimum	65.79% +/- 8.95% (micro average: 66.24%) (positive class: 0)
Maximum	65.79% +/- 8.95% (micro average: 66.24%) (positive class: 0)
Zero	65.79% +/- 8.95% (micro average: 66.24%) (positive class: 0)
Value	65.79% +/- 8.95% (micro average: 66.24%) (positive class: 0)
DuDeFiller	**69.49% +/- 5.65%** **(micro average: 69.50%) (positive class: 0)**

Table 24. The f-measure of "Naive Bayes" for DASD (DuDeFiller)

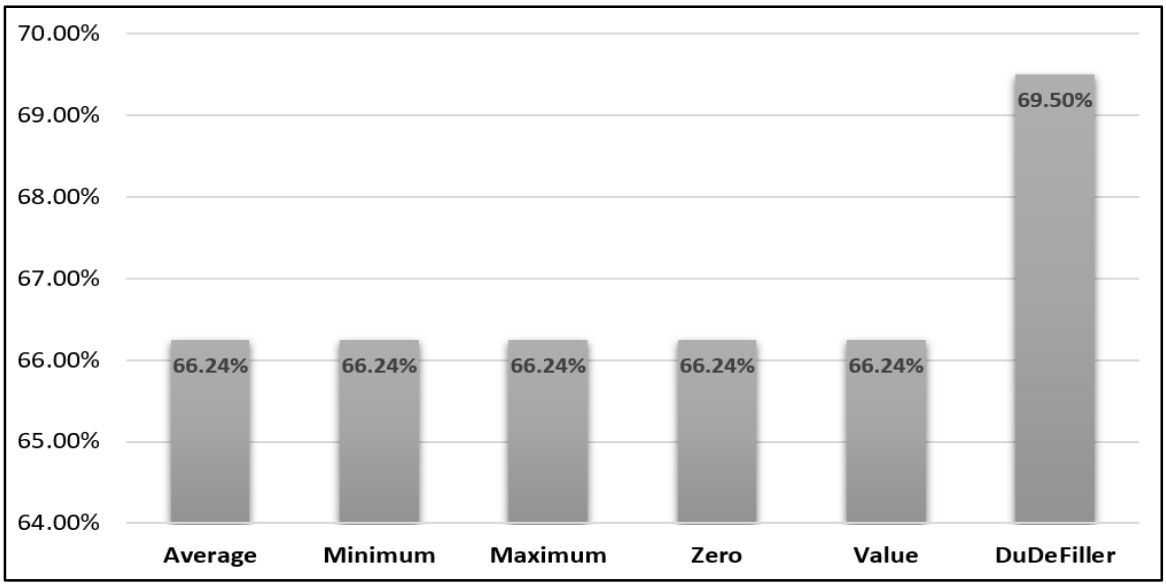

Fig. 64. The f-measure of "Naive Bayes" for DASD (DuDeFiller)

3) Deep Learning

The f-measure of the "Deep Learning" classifier for "Dresses Attribute Sales Dataset" (DASD), filled for missing values with six different algorithms, is shown in Table 25 and Fig. 65. The f-measure of all six missing value replacement algorithms, on DASD, are 67.85% for Average as a replacement, 67.85% for Minimum as a replacement, 67.85% for Maximum as a replacement, 67.85% for Zero as a replacement, 69.98% for Specific Value as a replacement and 71.13% for replacement with the DuDeFiller algorithm.

	F-measure
Average	67.37% +/- 7.00% (micro average: 67.85%) (positive class: 0)
Minimum	67.37% +/- 7.00% (micro average: 67.85%) (positive class: 0)
Maximum	67.37% +/- 7.00% (micro average: 67.85%) (positive class: 0)
Zero	67.37% +/- 7.00% (micro average: 67.85%) (positive class: 0)
Value	69.83% +/- 7.16% (micro average: 69.98%) (positive class: 0)
DuDeFiller	**71.14% +/- 4.30%** **(micro average: 71.13%) (positive class: 0)**

Table 25. The f-measure of "Deep Learning" for DASD (DuDeFiller)

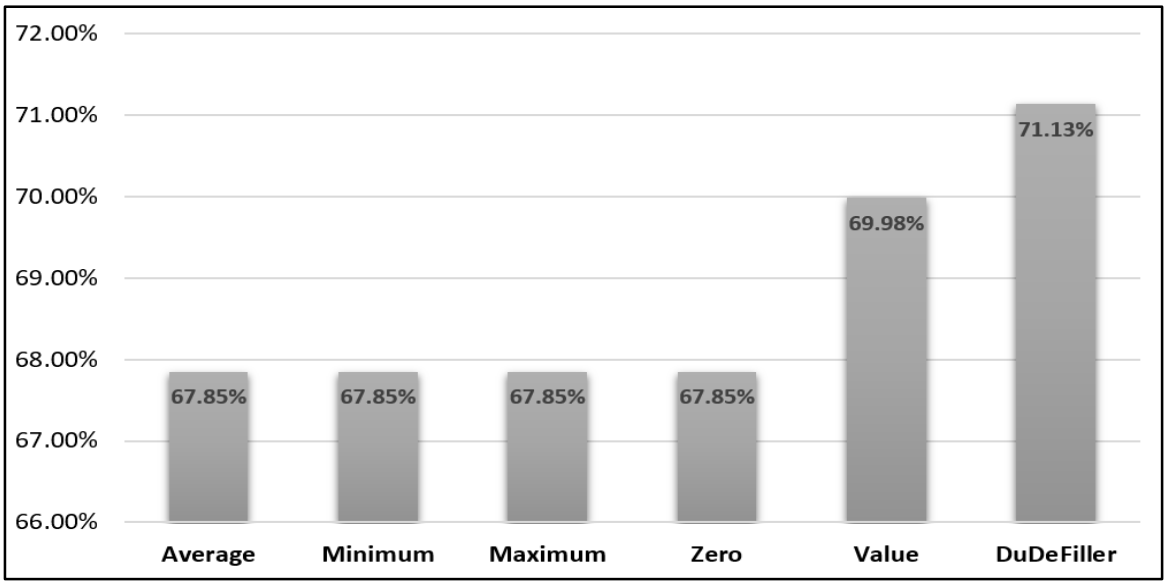

Fig. 65. The f-measure of "Deep Learning" for DASD (DuDeFiller)

4) Random Forest

The f-measure of the "Random Forest" classifier for "Dresses Attribute Sales Dataset" (DASD), filled for missing values with six different algorithms, is shown in Table 26 and Fig. 66. The f-measure of all six missing value replacement algorithms, on DASD, are 73.42% for Average as a replacement, 73.42% for Minimum as a replacement, 73.42% for Maximum as a replacement, 73.42% for Zero as a replacement, 73.42% for Specific Value as a replacement and 73.58% for replacement with the DuDeFiller algorithm.

	F-measure
Average	73.42% +/- 0.00% (micro average: 73.42%) (positive class: 0)
Minimum	73.42% +/- 0.00% (micro average: 73.42%) (positive class: 0)
Maximum	73.42% +/- 0.00% (micro average: 73.42%) (positive class: 0)
Zero	73.42% +/- 0.00% (micro average: 73.42%) (positive class: 0)
Value	73.42% +/- 0.00% (micro average: 73.42%) (positive class: 0)
DuDeFiller	**73.58% +/- 0.47%** **(micro average: 73.58%) (positive class: 0)**

Table 26. The f-measure of "Random Forest" for DASD (DuDeFiller)

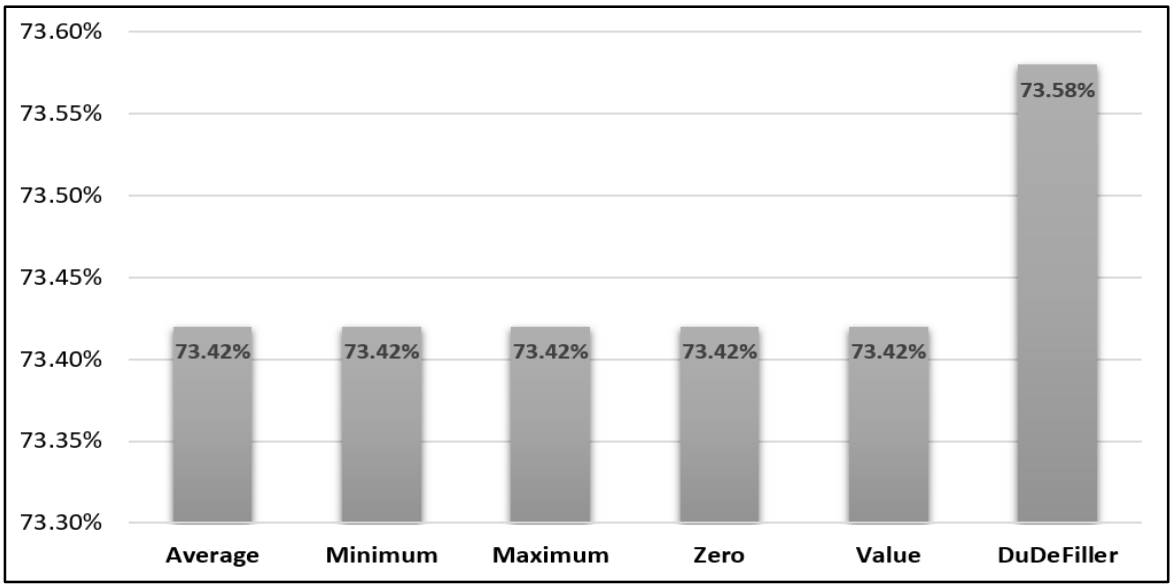

Fig. 66. The f-measure of "Random Forest" for DASD (DuDeFiller)

5) Logistic Regression

The f-measure of the "Logistic Regression" classifier for "Dresses Attribute Sales Dataset" (DASD), filled for missing values with six different algorithms, is shown in Table 27 and in Fig. 67. The f-measure of all six missing value replacement algorithms, on DASD, are 63.55% for Average as a replacement, 63.55% for Minimum as a replacement, 63.55% for Maximum as a replacement, 63.55% for Zero as a replacement, 63.55% for Specific Value as a replacement and 64.55% for replacement with the DuDeFiller algorithm.

	F-measure
Average	63.34% +/- 6.91% (micro average: 63.55%) (positive class: 0)
Minimum	63.34% +/- 6.91% (micro average: 63.55%) (positive class: 0)
Maximum	63.34% +/- 6.91% (micro average: 63.55%) (positive class: 0)
Zero	63.34% +/- 6.91% (micro average: 63.55%) (positive class: 0)
Value	63.34% +/- 6.91% (micro average: 63.55%) (positive class: 0)
DuDeFiller	**64.32% +/- 7.01%** **(micro average: 64.55%) (positive class: 0)**

Table 27. The f-measure of "Logistic Regression" for DASD (DuDeFiller)

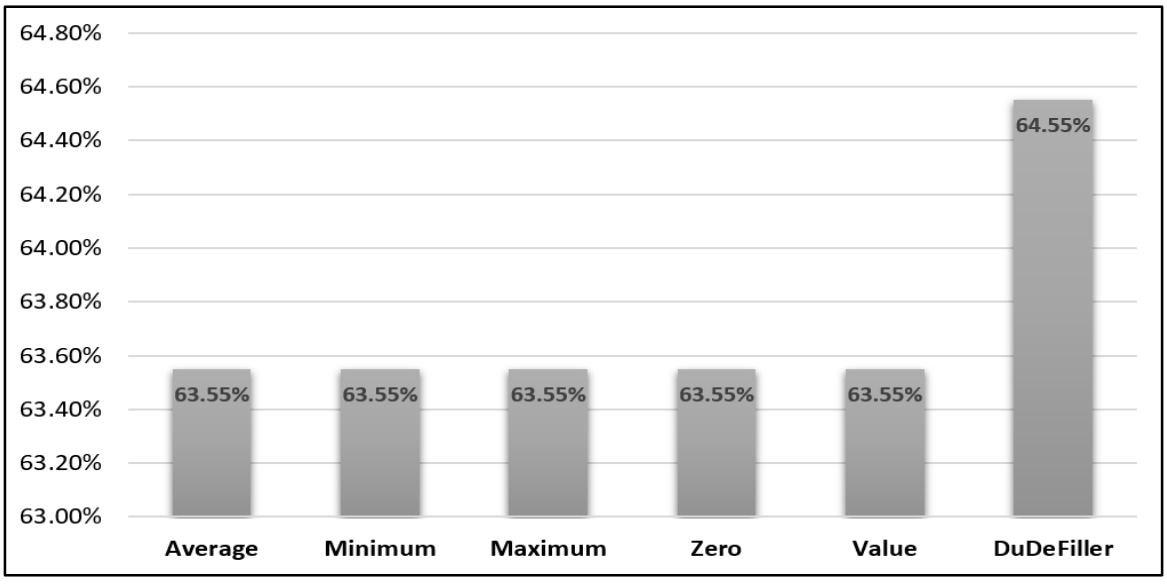

Fig. 67. The f-measure of "Logistic Regression" for DASD (DuDeFiller)

5.6. Evaluation of Results

Data Cleansing System (DCS) is assembled on “Duplicate Detection toolkit (DuDe)” [7] and this toolkit comprises the pool of duplicate detection procedures written in Java programming language. DuDe was established and verified by German data scientists over a period of time and constant perfections were made to advance its quality. The quality and the open source license of this DuDe kit fascinated us to get the assistance from it.

This work was started from the base point of DuDe kit because it was previously established and verified for its performance and as per software engineering principle, previously done work was reused and reinventing the wheel was avoided [92]. As per the “Quality in, quality out” [93] principle, the developed system is of high quality because the base and the input for the building block is of high quality.

Results of classification accuracy and f-measure are discussed in this section. These results are generated for the “Audiology” and “Dresses Attribute Sales” datasets filled for missing values with the proposed SimFiller and DuDeFiller algorithms along with other five missing values replacement algorithms. Experiments are conducted on the DuDeFiller algorithm for the duplicate fusion options because DuDeFuse and DuDeFiller are created on the similar mechanism so correctness of DuDeFiller also proves the correctness of the DuDeFuse algorithm. Duplicate fusion technique not only resolves the unintended duplicates issue but also helps to merge the deliberately created duplicates from the parallel sensors setup [49-52, 76, 77]. This technique is widely used for the fusion of multiple inputs received from the multiple sensors installed to record a similar event [50, 78].

The algorithms suggested in this study generates the replacement of missing and duplicate values by considering all the attributes of the record as a whole. Exiting replacement algorithms for missing values (average, minimum and maximum) independently select a single attribute and generate the replacement of missing value similar to the mechanism adopted by the “Naive Bayes” classifier. Considering this similarity, there is a chance to get the biases classification accuracy results of “Naive Bayes” classifier for the replicas filled with the existing (average, minimum and maximum) algorithms. Results show that the proposed algorithms improve the classification accuracy of selected classifiers including the classification accuracy of “Naive Bayes” classifier.

The approach used in the developed system is more rational and appropriate for several areas of life like in medicine field a patient medical history is matched with other patients as a whole and no medication is given to the similar patient in case of partial or non-match scenario.

5.7. Evaluation of Classification Accuracy Results (SimFiller for Audiology)

Classification accuracy results of "Decision Tree", "Naive Bayes", "Deep Learning" and "Random Forest" are evaluated in this section. These results are generated for the replicas of "Audiology" dataset filled for the missing values using a specific value, zero value, maximum value, minimum value, the average value and SimFiller algorithms. Classification accuracy improvement details of the selected classifiers are given in the section below:

1) Classification Accuracy Improvement of Decision Tree

Highest classification accuracy of Decision Tree is recorded for the SimFiller algorithm as 84.62% which is 7.70% higher with respect to the accuracy of other selected missing values filling algorithms. Classification accuracies of all six missing value replacement algorithms, on "Audiology" dataset, are 15.38% for Average as a replacement, 76.92% for Minimum as a replacement, 76.92% for Maximum as a replacement, 76.92% for Zero as a replacement, 15.38% for Specific Value as a replacement and 84.62% for replacement with the SimFiller algorithm as shown in Table 10 and Fig. 16.

2) Classification Accuracy Improvement of Naive Bayes

Highest classification accuracy of Naive Bayes is recorded for the SimFiller algorithm as 80.77% which is 11.54% higher with respect to the accuracy of other selected missing values filling algorithms. Classification accuracies of all six missing value replacement algorithms, on "Audiology" dataset, are 61.54% for Average as a replacement, 69.23% for Minimum as a replacement, 69.23% for Maximum as a replacement, 69.23% for Zero as a replacement, 65.38% for Specific Value as a replacement and 80.77% for replacement with the SimFiller algorithm as shown in Table 11 and Fig. 51.

3) Classification Accuracy Improvement of Deep Learning

Highest classification accuracy of Deep Learning is recorded for the SimFiller algorithm as 88.46% which is 11.54% higher with respect to the accuracy of other selected missing values filling algorithms. Classification accuracies of all six missing value

replacement algorithms, on "Audiology" dataset, are 76.92% for Average as a replacement, 73.08% for Minimum as a replacement, 73.08% for Maximum as a replacement, 73.08% for Zero as a replacement, 61.54% for Specific Value as a replacement and 88.46% for replacement with the SimFiller algorithm as shown in Table 12 and Fig. 52.

4) Classification Accuracy Improvement of Random Forest

Highest classification accuracy of Random Forest is recorded for the SimFiller algorithm as 61.54% which is 7.69% higher with respect to the accuracy of other selected missing values filling algorithms. Classification accuracies of all six missing value replacement algorithms, on "Audiology" dataset, are 53.85% for Average as a replacement, 50.00% for Minimum as a replacement, 50.00% for Maximum as a replacement, 50.00% for Zero as a replacement, 50.00% for Specific Value as a replacement and 61.54% for replacement with the SimFiller algorithm as shown in Table 13 and Fig. 53.

5.8. Evaluation of Classification Accuracy Results (DuDeFiller for Audiology)

Classification accuracy results of "Decision Tree", "Naive Bayes", "Deep Learning" and "Random Forest" are evaluated in this section. These results are generated for the replicas of "Audiology" dataset filled for the missing values using a specific value, zero value, maximum value, minimum value, the average value and DuDeFiller algorithms. Verification of the results is performed using the "using stratified sampling based on 10 folds' cross validation". Classification accuracy improvement details of the selected classifiers are given in the section below:

1) Classification Accuracy Improvement of Decision Tree

Highest classification accuracy of Decision Tree is recorded for the DuDeFiller algorithm as 69.03%. Classification accuracies of all six missing value replacement algorithms, on "Audiology" dataset, are 67.26% for Average as a replacement, 68.14% for Minimum as a replacement, 68.14% for Maximum as a replacement, 68.14% for Zero as a replacement, 68.14% for Specific Value as a replacement and 69.03% for replacement with the DuDeFiller algorithm as shown in Table 14 and Fig. 54.

2) Classification Accuracy Improvement of Naive Bayes

Highest classification accuracy of Naive Bayes is recorded including for the DuDeFiller algorithm as 80.97%. Classification accuracies of all six missing value replacement algorithms, on "Audiology" dataset, are 80.97% for Average as a replacement, 80.97% for Minimum as a replacement, 80.97% for Maximum as a replacement, 80.97% for Zero as a replacement, 80.53% for Specific Value as a replacement and 80.97% for replacement with the DuDeFiller algorithm as shown in Table 15 and Fig. 55.

3) Classification Accuracy Improvement of Deep Learning

Highest classification accuracy of Deep Learning is recorded for the DuDeFiller algorithm as 74.78%. Classification accuracies of all six missing value replacement algorithms, on "Audiology" dataset, are 74.34% for Average as a replacement, 70.80% for Minimum as a replacement, 70.80% for Maximum as a replacement, 70.80% for Zero as a replacement, 73.45% for Specific Value as a replacement and 74.78% for replacement with the DuDeFiller algorithm as shown in Table 16 and Fig. 56.

4) Classification Accuracy Improvement of Random Forest

Highest classification accuracy of Random Forest is recorded for the DuDeFiller algorithm as 75.22%. Classification accuracies of all six missing value replacement algorithms, on "Audiology" dataset, are 71.68% for Average as a replacement, 73.01% for Minimum as a replacement, 73.01% for Maximum as a replacement, 73.01% for Zero as a replacement, 73.01% for Specific Value as a replacement and 75.22% for replacement with the DuDeFiller algorithm as shown in Table 17 and Fig. 57.

5.9. Evaluation of Classification Accuracy Results (DuDeFiller for DASD)

Classification accuracy results of "Decision Tree", "Naive Bayes", "Deep Learning", "Random Forest" and "Logistic Regression" are evaluated in this section. These results are generated for the replicas of "Dresses Attribute Sales Dataset" (DASD) filled for the missing values using a specific value, zero value, maximum value, minimum value, the average value and DuDeFiller algorithms. Verification of the results is performed using the "using stratified sampling based on 10 folds' cross validation". Classification accuracy improvement details of the selected classifiers are given in the section below:

1) Classification Accuracy Improvement of Decision Tree

Highest classification accuracy of Decision Tree is recorded for the DuDeFiller algorithm as 58.20%. Classification accuracies of all six missing value replacement algorithms, on “Dresses Attribute Sales Dataset” (DASD), are 58.00% for Average as replacement, 58.00% for Minimum as replacement, 58.00% for Maximum as replacement, 58.00% for Zero as replacement, 58.00% for Specific Value as replacement and 58.20% for replacement with DuDeFiller algorithm as shown in Table 18 and Fig. 58.

2) Classification Accuracy Improvement of Naive Bayes

Highest classification accuracy of Naive Bayes is recorded for the DuDeFiller algorithm as 61.20%. Classification accuracies of all six missing value replacement algorithms, on “Dresses Attribute Sales Dataset” (DASD), are 58.20% for Average as replacement, 58.20% for Minimum as replacement, 58.20% for Maximum as replacement, 58.20% for Zero as replacement, 58.20% for Specific Value as replacement and 61.20% for replacement with DuDeFiller algorithm as shown in Table 19 and Fig. 59.

3) Classification Accuracy Improvement of Deep Learning

Highest classification accuracy of Deep Learning is recorded for the DuDeFiller algorithm as 63.80%. Classification accuracies of all six missing value replacement algorithms, on “Dresses Attribute Sales Dataset” (DASD), are 60.20% for Average as a replacement, 60.20% for Minimum as a replacement, 60.20% for Maximum as a replacement, 60.20% for Zero as a replacement, 61.40% for Specific Value as a replacement and 63.80% for replacement with the DuDeFiller algorithm as shown in Table 20 and Fig. 60.

4) Classification Accuracy Improvement of Random Forest

Highest classification accuracy of Random Forest is recorded for the DuDeFiller algorithm as 75.22%. Classification accuracies of all six missing value replacement algorithms, on “Dresses Attribute Sales Dataset” (DASD), are 58.00% for Average as replacement, 58.00% for Minimum as replacement, 58.00% for Maximum as replacement, 58.00% for Zero as replacement, 58.00% for Specific Value as replacement and 58.20% for replacement with DuDeFiller algorithm as shown in Table 21 and Fig. 61.

5) Classification Accuracy Improvement of Logistic Regression

Highest classification accuracy of Logistic Regression is recorded for the DuDeFiller algorithm as 75.22%. Classification accuracies of all six missing value replacement algorithms, on "Dresses Attribute Sales Dataset" (DASD), are 56.40% for Average as replacement, 56.40% for Minimum as replacement, 56.40% for Maximum as replacement, 56.40% for Zero as replacement, 56.40% for Specific Value as replacement and 57.60% for replacement with DuDeFiller algorithm as shown in Table 22 and Fig. 62.

5.10. Evaluation of F-measure Results (DuDeFiller for DASD)

F-measure results "Decision Tree", "Naive Bayes", "Deep Learning", "Random Forest" and "Logistic Regression" are evaluated in this section. These results are generated for the replicas of "Dresses Attribute Sales Dataset" (DASD) filled for the missing values using a specific value, zero value, maximum value, minimum value, the average value and DuDeFiller algorithms. Verification of the results is performed using the "using stratified sampling based on 10 folds' cross validation. F-measure improvement details of the selected classifiers are given in the section below:

1) F-measure Improvement of Decision Tree

Highest f-measure of Decision Tree is recorded for the DuDeFiller algorithm as 73.58%. The f-measure of all six missing value replacement algorithms, on "Dresses Attribute Sales Dataset" (DASD), are 73.42% for Average as replacement, 73.42% for Minimum as replacement, 73.42% for Maximum as replacement, 73.42% for Zero as replacement, 73.42% for Specific Value as replacement and 73.58% for replacement with DuDeFiller algorithm as shown in Table 23 and Fig. 63.

2) F-measure Improvement of Naive Bayes

Highest f-measure of Naive Bayes is recorded for the DuDeFiller algorithm as 69.50%. The f-measure of all six missing value replacement algorithms, on "Dresses Attribute Sales Dataset" (DASD), are 66.24% for Average as replacement, 66.24% for Minimum as replacement, 66.24% for Maximum as replacement, 66.24% for Zero as replacement, 66.24% for Specific Value as replacement and 69.50% for replacement with DuDeFiller algorithm as shown in Table 24 and Fig. 64.

3) F-measure Improvement of Deep Learning

Highest f-measure of Deep Learning is recorded for the DuDeFiller algorithm as 71.13%. The f-measure of all six missing value replacement algorithms, on “Dresses Attribute Sales Dataset” (DASD), are 67.85% for Average as replacement, 67.85% for Minimum as replacement, 67.85% for Maximum as replacement, 67.85% for Zero as replacement, 69.98% for Specific Value as replacement and 71.13% for replacement with DuDeFiller algorithm as shown in Table 25 and Fig. 65.

4) F-measure Improvement of Random Forest

Highest f-measure of Random Forest is recorded for the DuDeFiller algorithm as 73.58%. The f-measure of all six missing value replacement algorithms, on “Dresses Attribute Sales Dataset” (DASD), are 73.42% for Average as replacement, 73.42% for Minimum as replacement, 73.42% for Maximum as replacement, 73.42% for Zero as replacement, 73.42% for Specific Value as replacement and 73.58% for replacement with DuDeFiller algorithm as shown in Table 26 and Fig. 66.

5) F-measure Improvement of Logistic Regression

Highest f-measure of Logistic Regression is recorded for the DuDeFiller algorithm as 64.55%. The f-measure of all six missing value replacement algorithms, on “Dresses Attribute Sales Dataset” (DASD), are 63.55% for Average as replacement, 63.55% for Minimum as replacement, 63.55% for Maximum as replacement, 63.55% for Zero as replacement, 63.55% for Specific Value as replacement and 64.55% for replacement with DuDeFiller algorithm as shown in Table 27 and in the Fig. 67.

Chapter 6

CONCLUSION AND FUTURE WORK

This chapter discusses the outcomes and the impact of the proposed study and concludes the overall benefit and result from this study. This chapter also defines the future directions and advises the prospect research from the presented idea in the earlier chapters.

6.1. Conclusion

Sensors, devices and human-generated data volumes encompass data irregularities containing missing and duplicate records. Volumes of data are expanding every second because sources of data creation are also expanding. Computerized data cleansing is performed by filling the missing value and merging the duplicate records because manual fixation of data irregularities is not possible due to massive volumes of data. Automatic cleansing methods recover the quality of the dataset before applying any data mining process on it to extract some useful information.

This research develops an efficient duplicate data detection based cleansing system named Data Cleansing System (DCS) to fix the inconsistencies of any dataset. Data Cleansing System (DCS) has four subsystems named as Pre-Processing Subsystem (PPS), Duplicate Detection Subsystem (DDS), Duplicate Fusion Subsystem (DFS) and Post-Operation Subsystem (POS).

Pre-Processing Subsystem deals with the preliminary activities executed on the input file before performing any data cleansing task on it. PPS is further divided into four main modules. A user interface is created with an option to select and load the CSV file in the subsystem. After successful loading, the subsystem reads the input dataset from CSV file, extracts the list of attributes from it. The subsystem then asks the user to select the required attributes from the list of attributes and the selected attributes are used to extract the required data from the dataset which is returned to the main system for further processing.

After the successful preprocessing, extracted data and the attributes list is passed to the Duplicate Detection Subsystem (DDS) to find duplicates from it. Duplicate Detection Subsystem (DDS) is further divided into four main modules. In the first module, the user is asked to set the similarity threshold after successful data extraction then records pairs are

generated and their similarity is calculated. If pairs similarity exceeds the similarity threshold, pairs are saved for the later processing.

In Duplicate Fusion Subsystem (DFS), extracted duplicate pairs are further analyzed to merge them into a single record. Duplicate fusion is one of the duplicate cleansing process developed to clean the duplicate records from the datasets. During the data fusion, duplicate records are merged to create a new and more accurate and useful record as a result. Duplicate fusion technique not only resolves the unintended duplicates issue but also helps to merge the deliberately created duplicates from the parallel sensors setup. This technique is widely used for the fusion of multiple inputs received from the multiple sensors installed to record a similar event.

Post-Operation Subsystem (POS) deals with the activities executed to create the output file after performing a cleansing task on it during the Duplicate Fusion Subsystem (DFS). DCS passes the output of the Data Fusion Subsystem (DFS) to the Post-Operation Subsystem (POS) in the form of fused duplicates which is further processed and written to the file for the user.

The raw dataset file is given as input to the system which is sent to the Pre-Processing Subsystem (PPS) to extract and select the attributes from the dataset. The data is extracted for the selected attributes which are then passed to the Duplicate Detection Subsystem (DDS) to extract the duplicate pairs from the dataset. The extracted duplicate pairs are then passed to the Duplicate Fusion Subsystem (DFS) to fuse the duplicates into a single record. The fused duplicates are then passed to the Post-Operation Subsystem (POS) to create the cleaned copy of input dataset by adding attribute names and non-duplicate data in it.

Three new algorithms named SimFiller, DuDeFiller and DuDeFuse are also introduced for the Duplicate Fusion Subsystem (DFS). SimFiller is similarity-based missing values algorithm which clears the missing value of one record by getting the value of the attribute from the most similar record. DuDeFiller is duplicate detection based missing values algorithm which clears the missing value of one record by getting the value of the attribute from its duplicate record.

DuDeFuse is duplicate detection and fusion algorithm which accepts input from the Duplicate Detection Subsystem as duplicate record pairs. DuDeFuse calculates the similarity and occurrence of each possible value of an attribute for each of the records within the duplicate pair under consideration.

Experimentations accomplished during this research attested that Data Cleansing System (DCS) is a significant, effective and suitable system to fix the inconsistencies of any dataset. Datasets fixed for the inconsistencies with the system conveys upmost f-measure and accuracy for all the selected datasets within all other selected data cleansing methods.

The approach used in the developed system is more rational and appropriate for several areas of life like in medicine field a patient medical history is matched with other patients as a whole and no medication is given to the similar patient in case of partial or non-match scenario.

6.2. Contributions to the Data Science Body of Knowledge

A fully customized data cleansing system is developed by this research which is a huge addition to the data science body of knowledge. Following contributions are also added by this research work:

- ☑ Two efficient and robust similarity-based and duplicate detection based missing values filling algorithms are added to the data science body of knowledge those fill the dataset's missing values more efficiently than the existing algorithms.

- ☑ A proficient and robust duplicate detection based data fusion algorithm is added to the data science body of knowledge which fills the dataset's missing values more efficiently than the existing algorithms.

6.3. Future Work

Similarity threshold was set to a particular value for the tests accomplished during this research and it is also observed that accuracy is also changed by changing the value of the similarity threshold. Finding the similarity threshold and its relationship with the accuracy of each dataset are out of the scope of this research but considered as strong future direction for this research. work.

Data Cleansing System (DCS) can also be extended in terms of its subsystems to add a new subsystem or in terms of algorithms to add new algorithms in existing subsystems to resolve the other irregularities of datasets including outliers of the datasets.

REFERENCES

1. Kim, J. and J.H. Ryu, Quantifying a Threshold of Missing Values for Gap Filling Processes in Daily Precipitation Series. Water Resources Management, 2015. 29(11): p. 4173-4184.

2. Bingwei, H., et al. A new method for filling missing values by gray relational analysis. in 2011 2nd International Conference on Artificial Intelligence, Management Science and Electronic Commerce (AIMSEC). 2011.

3. Lee, K.C., et al. Missing Value Estimation Based on Dynamic Attribute Selection. in Knowledge Discovery and Data Mining. Current Issues and New Applications. 2000. Berlin, Heidelberg: Springer Berlin Heidelberg.

4. Zhang, S., et al., Missing Value Imputation Based on Data Clustering, in Transactions on Computational Science I, M.L. Gavrilova and C.J.K. Tan, Editors. 2008, Springer Berlin Heidelberg: Berlin, Heidelberg. p. 128-138.

5. Varnild, E., Replace Missing Values. RapidMiner Studio Core, 2018. 8.1.

6. Kotsiantis, S., et al. Filling missing temperature values in weather data banks. in 2006 2nd IET International Conference on Intelligent Environments - IE 06. 2006.

7. Draisbach, U. and F.N. Hasso, DuDe: The Duplicate Detection Toolkit. hpi.de/ naumann/ sites/ dude/ javadoc, 2012.

8. Rekatsinas, T., X.L. Dong, and D. Srivastava, Characterizing and selecting fresh data sources, in Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data. 2014, ACM: Snowbird, Utah, USA. p. 919-930.

9. Ragel, A. Preprocessing of missing values using robust association rules. in Principles of Data Mining and Knowledge Discovery. 1998. Berlin, Heidelberg: Springer Berlin Heidelberg.

10. Newcombe H. B., K.J.M., Axford S. J, and James A. P., Automatic linkage of vital records. Science. 1959. vol. 130: p. 954–959.

11. Susanti, S.P. and F.N. Azizah. Imputation of missing value using dynamic Bayesian network for multivariate time series data. in 2017 International Conference on Data and Software Engineering (ICoDSE). 2017.

12. Lobo, O.O. and M. Numao. Ordered Estimation of Missing Values. in Methodologies for Knowledge Discovery and Data Mining. 1999. Berlin, Heidelberg: Springer Berlin Heidelberg.

13. Liu, W.Z., et al. Techniques for dealing with missing values in classification. in Advances in Intelligent Data Analysis Reasoning about Data. 1997. Berlin, Heidelberg: Springer Berlin Heidelberg.

14. Honghai, F., et al. A SVM Regression Based Approach to Filling in Missing Values. in Knowledge-Based Intelligent Information and Engineering Systems. 2005. Berlin, Heidelberg: Springer Berlin Heidelberg.

15. Ralescu, A. and S. Visa. On filling-in missing attribute values for Bayes and fuzzy classifiers. in NAFIPS 2008 - 2008 Annual Meeting of the North American Fuzzy Information Processing Society. 2008.

16. Manikanta, C. and V. Mamatha Jadav. Evaluation of modified PLS regression method to fill the missing values in training dataset. in 2015 International Conference on Smart Sensors and Systems (IC-SSS). 2015.

17. Pérez, A., et al., Use of the mean, hot deck and multiple imputation techniques to predict outcome in intensive care unit patients in Colombia. Statistics in Medicine, 2002. 21(24): p. 3885-3896.

18. Barakat, M.S., et al., The effect of imputing missing clinical attribute values on training lung cancer survival prediction model performance. Health Information Science and Systems, 2017. 5(1): p. 16.

19. Salzberg, S.L., C4.5: Programs for Machine Learning by J. Ross Quinlan. Morgan Kaufmann Publishers, Inc., 1993. Machine Learning, 1994. 16(3): p. 235-240.

20. Zhang, C. and S. Zhang, Association rule mining: models and algorithms. 2002: Springer-Verlag. 228.

21. Zhao-hong, W. Numeric Missing Value's Hot Deck Imputation Based on Cloud Model and Association Rules. in 2010 Second International Workshop on Education Technology and Computer Science. 2010.

22. Surong, J., L. Jiangqiao, and Y. Yuhai. A Data Analysis Algorithm of Missing Point Association Rules for Air Target. in 2015 14th International Symposium on Distributed Computing and Applications for Business Engineering and Science (DCABES). 2015.

23. Xu, D. and Y. Tian, A Comprehensive Survey of Clustering Algorithms. Annals of Data Science, 2015. 2(2): p. 165-193.

24. Ji, Y., W. Hong, and J. Qi. Missing Value Prediction Using Co-clustering and RBF for Collaborative Filtering. in 2015 International Conference on Cloud Computing and Big Data (CCBD). 2015.

25. Wang, L., et al., Modelling method with missing values based on clustering and support vector regression. Journal of Systems Engineering and Electronics, 2010. 21(1): p. 142-147.

26. Albayrak, M., K. Turhan, and B. Kurt. A missing data imputation approach using clustering and maximum likelihood estimation. in 2017 Medical Technologies National Congress (TIPTEKNO). 2017.

27. Mohammadi, A. and M.H. Saraee. Estimating Missing Value in Microarray Data Using Fuzzy Clustering and Gene Ontology. in 2008 IEEE International Conference on Bioinformatics and Biomedicine. 2008.

28. Zanaty, E.A., Support Vector Machines (SVMs) versus Multilayer Perception (MLP) in data classification. Egyptian Informatics Journal, 2012. 13(3): p. 177-183.

29. Anderson, H.S. and M.R. Gupta. Expected kernel for missing features in support vector machines. in 2011 IEEE Statistical Signal Processing Workshop (SSP). 2011.

30. Kim, J., et al., Deep neural networks with weighted spikes. Neurocomputing, 2018. 311: p. 373-386.

31. Setiawan, N.A., P.A. Venkatachalam, and A.F.M. Hani. Missing Attribute Value Prediction Based on Artificial Neural Network and Rough Set Theory. in 2008 International Conference on BioMedical Engineering and Informatics. 2008.

32. Sobrevilla, K.L.M.D., et al. Daily weather forecast in Tiwi, Albay, Philippines using Artificial Neural Network with missing values Imputation. in 2016 IEEE Region 10 Conference (TENCON). 2016.

33. Kramer, O., K-Nearest Neighbors, in Dimensionality Reduction with Unsupervised Nearest Neighbors, O. Kramer, Editor. 2013, Springer Berlin Heidelberg: Berlin, Heidelberg. p. 13-23.

34. Meesad, P. and K. Hengpraprohm. Combination of KNN-Based Feature Selection and KNNBased Missing-Value Imputation of Microarray Data. in 2008 3rd International Conference on Innovative Computing Information and Control. 2008.

35. Sen, S., M.N. Das, and R. Chatterjee. A Weighted kNN approach to estimate missing values. in 2016 3rd International Conference on Signal Processing and Integrated Networks (SPIN). 2016.

36. Cai, Q., et al. The research of missing value estimation of gene sequence based on improved KNN. in 2009 4th International Conference on Computer Science & Education. 2009.

37. Keerin, P., W. Kurutach, and T. Boongoen. Cluster-based KNN missing value imputation for DNA microarray data. in 2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC). 2012.

38. Zhu, M. and C. Xingbing. Iterative KNN imputation based on GRA for missing values in TPLMS. in 2015 4th International Conference on Computer Science and Network Technology (ICCSNT). 2015.

39. Fürnkranz, J., Decision Tree, in Encyclopedia of Machine Learning and Data Mining, C. Sammut and G.I. Webb, Editors. 2016, Springer US: Boston, MA. p. 1-5.

40. Ming-Fen, W. and W. Ting-Liang. An algorithm: Filling the missing values in the incomplete decision table. in 2008 International Conference on Machine Learning and Cybernetics. 2008.

41. Gavankar, S. and S. Sawarkar. Decision Tree: Review of Techniques for Missing Values at Training, Testing and Compatibility. in 2015 3rd International Conference on Artificial Intelligence, Modelling and Simulation (AIMS). 2015.

42. Webb, G.I., Naïve Bayes, in Encyclopedia of Machine Learning, C. Sammut and G.I. Webb, Editors. 2010, Springer US: Boston, MA. p. 713-714.

43. Baharim, K.N., M.S. Kamaruddin, and F. Jusof. Leveraging Missing Values in Call Detail Record Using Naïve Bayes for Fraud Analysis. in 2008 International Conference on Information Networking. 2008.

44. Gupta, A.K. and N. Sardana. Naïve Bayes Approach for Predicting Missing Links in Ego Networks. in 2016 IEEE International Symposium on Nanoelectronic and Information Systems (iNIS). 2016.

45. Menezes, S.L. and G. Varkey. Prediction of Missing Items Using Naive Bayes Classifier and Graph Based Prediction. in 2013 Third International Conference on Advances in Computing and Communications. 2013.

46. Bilke, A., et al., Automatic data fusion with HumMer, in Proceedings of the 31st international conference on Very large data bases. 2005, VLDB Endowment: Trondheim, Norway. p. 1251-1254.

47. Banerjee, S., et al. A fusion approach for classifying duplicate problem reports. in 2013 IEEE 24th International Symposium on Software Reliability Engineering (ISSRE). 2013.

48. Bleiholder, J., K. Draba, and F. Naumann, FuSem: exploring different semantics of data fusion, in Proceedings of the 33rd international conference on Very large data bases. 2007, VLDB Endowment: Vienna, Austria. p. 1350-1353.

49. Andr, #233, and Bolles, A flexible framework for multisensor data fusion using data stream management technologies, in Proceedings of the 2009 EDBT/ICDT Workshops. 2009, ACM: Saint-Petersburg, Russia. p. 193-200.

50. Bader, K., B. Lussier, and W. Schön, A fault tolerant architecture for data fusion: A real application of Kalman filters for mobile robot localization. Robotics and Autonomous Systems, 2017. 88(Supplement C): p. 11-23.

51. Khaleghi, B., et al., Multisensor data fusion: A review of the state-of-the-art. Information Fusion, 2013. 14(1): p. 28-44.

52. Alyannezhadi, M.M., A.A. Pouyan, and V. Abolghasemi, An efficient algorithm for multisensory data fusion under uncertainty condition. Journal of Electrical Systems and Information Technology, 2017. 4(1): p. 269-278.

53. Haroun, A., et al., Data fusion in automotive applications. Personal Ubiquitous Comput., 2017. 21(3): p. 443-455.

54. Rezig, E.K., et al. ORLF: A flexible framework for online record linkage and fusion. in 2016 IEEE 32nd International Conference on Data Engineering (ICDE). 2016.

55. Gruenheid, A., X.L. Dong, and D. Srivastava, Incremental record linkage. Proc. VLDB Endow., 2014. 7(9): p. 697-708.

56. Dong, X.L., et al., Knowledge-based trust: estimating the trustworthiness of web sources. Proc. VLDB Endow., 2015. 8(9): p. 938-949.

57. Pohl, M., Naive Duplicate Detection (Class). hpi.de/ naumann/ sites/ dude/ javadoc, 2012.

58. Lindenberg, F., Duplicate Count SNM (Class). hpi.de/ naumann/ sites/ dude/ javadoc, 2012.

59. Pohl, M., Sorted Neighborhood Method (Class). hpi.de/ naumann/ sites/ dude/ javadoc, 2012.

60. Whang, S.E., et al., Entity Resolution with Iterative Blocking in SIGMOD 2009. 2009, Stanford: Providence, Rhode Island.

61. Thomas, F., Lego (Class). hpi.de/ naumann/ sites/ dude/ javadoc, 2012.

62. Dyck, J., GSwoosh (Class). hpi.de/ naumann/ sites/ dude/ javadoc, 2012.

63. Dyck, J., RSwoosh (Class). hpi.de/ naumann/ sites/ dude/ javadoc, 2012.

64. Pohl, M. and U. Draisbach, Naive Blocking Algorithm (Class). hpi.de/ naumann/ sites/ dude/ javadoc, 2012.

65. Abedjan, Z., A. Heise, and M. Pohl, Levenshtein Distance Function (Class). hpi.de/ naumann/ sites/ dude/ javadoc, 2012.

66. Abedjan, Z., A. Heise, and M. Pohl, Euclidean Distance Function (Class). hpi.de/ naumann/ sites/ dude/ javadoc, 2012.

67. Abedjan, Z., A. Heise, and M. Pohl, Jaro Winkler Function (Class). hpi.de/ naumann/ sites/ dude/ javadoc, 2012.

68. Abedjan, Z. and A. Heise, Jaccard Similarity Function (Class). hpi.de/ naumann/ sites/ dude/ javadoc, 2012.

69. Abedjan, Z., A. Heise, and M. Pohl, Cosine Similarity Function (Class). hpi.de/ naumann/ sites/ dude/ javadoc, 2012.

70. Abedjan, Z., A. Heise, and M. Pohl, Block Distance Function (Class). hpi.de/ naumann/ sites/ dude/ javadoc, 2012.

71. Abedjan, Z., A. Heise, and M. Pohl, Dice Coefficient Function (Class). hpi.de/ naumann/ sites/ dude/ javadoc, 2012.

72. Agarwala, R., Data Modeling, ETL, and Data Quality Guide. Oracle® Warehouse Builder, 2011. 11.2(E10935-05): p. 573-598.

73. Rezig, E.K., et al. Query-time record linkage and fusion over Web databases. in 2015 IEEE 31st International Conference on Data Engineering. 2015.

74. Varnild, E., Remove Duplicates. RapidMiner Studio Core, 2018. 8.1.

75. Bleiholder, J. and F. Naumann, Data fusion. ACM Comput. Surv., 2009. 41(1): p. 1-41.

76. Pinto, A.R., et al., An approach to implement data fusion techniques in wireless sensor networks using genetic machine learning algorithms. Information Fusion, 2014. 15(Supplement C): p. 90-101.

77. Rawat, S. and S. Rawat, Multi-sensor data fusion by a hybrid methodology – A comparative study. Computers in Industry, 2016. 75(Supplement C): p. 27-34.

78. Kubelka, V., M. Reinstein, and T. Svoboda, Improving multimodal data fusion for mobile robots by trajectory smoothing. Robotics and Autonomous Systems, 2016. 84(Supplement C): p. 88-96.

79. Bronselaer, A., D. Van Britsom, and G. De Tré, Pointwise multi-valued fusion. Information Fusion, 2015. 25(Supplement C): p. 121-133.

80. Dong, X.L., L. Berti-Equille, and D. Srivastava, Data Fusion: Resolving Conflicts from Multiple Sources, in Web-Age Information Management: 14th International Conference, WAIM 2013, Beidaihe, China, June 14-16, 2013. Proceedings, J. Wang, et al., Editors. 2013, Springer Berlin Heidelberg: Berlin, Heidelberg. p. 64-76.

81. Dong, X.L., et al., From data fusion to knowledge fusion. Proc. VLDB Endow., 2014. 7(10): p. 881-892.

82. Wu, S. and F. Crestani, A geometric framework for data fusion in information retrieval. Information Systems, 2015. 50(Supplement C): p. 20-35.

83. Peña Saldarriaga, S., E. Morin, and C. Viard-Gaudin, Ranking Fusion Methods Applied to On-Line Handwriting Information Retrieval, in Advances in Information Retrieval: 32nd European Conference on IR Research, ECIR 2010, Milton Keynes, UK, March 28-31, 2010.Proceedings, C. Gurrin, et al., Editors. 2010, Springer Berlin Heidelberg: Berlin, Heidelberg. p. 253-264.

84. Pochampally, R., et al., Fusing data with correlations, in Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data. 2014, ACM: Snowbird, Utah, USA. p. 433-444.

85. Velic, M., T. Grzinic, and I. Padavic. Wisdom of crowds algorithm for stock market predictions. in Proceedings of the ITI 2013 35th International Conference on Information Technology Interfaces. 2013.

86. Kattan, M.W., et al., The Wisdom of Crowds of Doctors:Their Average Predictions Outperform Their Individual Ones. Medical Decision Making, 2016. 36(4): p. 536-540.

87. Paltoglou, G., M. Salampasis, and M. Satratzemi, Simple Adaptations of Data Fusion Algorithms for Source Selection, in Advances in Information Retrieval: 31th European Conference on IR Research, ECIR 2009, Toulouse, France, April 6-9, 2009. Proceedings, M. Boughanem, et al., Editors. 2009, Springer Berlin Heidelberg: Berlin, Heidelberg. p. 497-508.

88. Dorman, C. and V. Rajlich. Software Change in the Solo Iterative Process: An Experience Report. in 2012 Agile Conference. 2012.

89. Lui, K.M. and K.C.C. Chan. Software Process Fusion: Uniting Pair Programming and Solo Programming Processes. in Software Process Change. 2006. Berlin, Heidelberg: Springer Berlin Heidelberg.

90. Galkina, O. and V. Yachenko. Application of iterative software development methodologies for reducing service quality gaps. in Proceedings of the 2014 IEEE NW Russia Young Researchers in Electrical and Electronic Engineering Conference. 2014.

91. Macek, O. and M. Komárek. The practical method of motivating students to iterative software development. in 2011 24th IEEE-CS Conference on Software Engineering Education and Training (CSEE&T). 2011.

92. Schacht, S. and A. Mädche. How to Prevent Reinventing the Wheel? – Design Principles for Project Knowledge Management Systems. in Design Science at the Intersection of Physical and Virtual Design. 2013. Berlin, Heidelberg: Springer Berlin Heidelberg.

93. Barenfanger, J., Quality in, quality out: rejection criteria and guidelines for commonly (mis)used tests. Clinical Microbiology Newsletter, 2000. 22(9): p. 65-72.

94. Quinlan, R., Audiology (Standardized) Data Set UCI Machine Learning Repository, 1992.

95. Usman, M. and A. Ahmed, Dresses Attribute Sales Data Set. UCI Machine Learning Repository, 2014.

96. Varnild, E., Cross Validation. RapidMiner Studio Core, 2018. 8.1.

97. Sokolova, M., N. Japkowicz, and S. Szpakowicz. Beyond Accuracy, F-Score and ROC: A Family of Discriminant Measures for Performance Evaluation. in AI 2006: Advances in Artificial Intelligence. 2006. Berlin, Heidelberg: Springer Berlin Heidelberg.