

# CLEANING AND CLUSTERING OF SENSOR DATA BY K-MEANS ALGORITHM FOR EFFICIENT QUERY PROCESSING

Submitted by

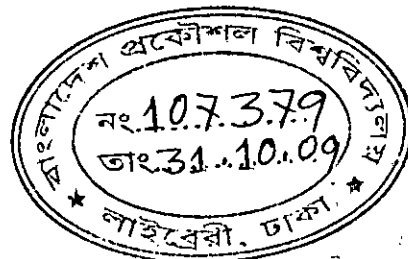
**MD. MUHIDUL ISLAM KHAN**  
Student ID: 100705049P

A report submitted to the Department of Computer Science and Engineering in partial fulfillment of the requirements for the degree of Masters of Engineering in Computer Science and Engineering

Supervised by

**Dr. A. S. M. Latiful Hoque**  
Associate Professor, Department of CSE, BUET

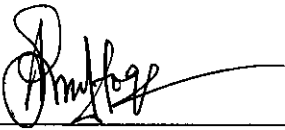
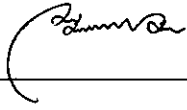
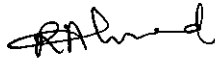
COMPUTER SCIENCE AND ENGINEERING  
BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY, DHAKA  
AUGUST, 2009



#107379#

The project "CLEANING AND CLUSTERING OF SENSOR DATA BY K-MEANS ALGORITHM FOR EFFICIENT QUERY PROCESSING", submitted by Md. Muhidul Islam Khan, Roll No. 100705049P, Session April 2009, to the Department of Computer Science and Engineering, has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of Master of Engineering in (Computer Science and Engineering) and approved as to its style and contents. Examination held on 23/08/09.

**Board of Examiners**

1.   
\_\_\_\_\_ Chairman  
(Supervisor)  
Dr. Abu Sayed Md. Latiful Hoque  
Associate Professor  
Department of CSE  
BUET, Dhaka-1000
  
2.   
\_\_\_\_\_ Member  
Dr. Md. Mostofa Akbar  
Associate Professor  
Department of CSE  
BUET, Dhaka-1000
  
3.   
\_\_\_\_\_ Member  
Dr. Reaz Ahmed  
Assistant Professor  
Department of CSE  
BUET, Dhaka-1000

## DECLARATION

I, hereby, declare that the work presented in this thesis is the out come of the investigation performed by me under the supervisor of Dr. A. S. M. Latiful Hoque, Associate Professor, Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, Dhaka. I also declare that no part of this thesis and thereof has been or is being submitted elsewhere for the award of any degree or diploma.

Signature

*M. Islam*

Md. Muhiul Islam Khan

## Abstract

The way of collecting sensor data will face a revolution when the newly developing technology of distributed sensor networks becomes fully functional and widely available. Distributed sensor networks are indeed an attractive technology, but the program/stack memory and the battery life of today nodes do not enable complex data mining in runtime. Effective data mining can be implemented on the central base station, where the computational power is not generally constrained. Today's real-world databases are highly susceptible to noisy, missing and inconsistent data because of their typically huge size and their likely origin from multiple, heterogeneous sources. Low-quality data will lead to low-quality mining results.

There are many possible reasons for noisy data (having incorrect attribute values). The data collection sensor nodes used may be faulty. Errors in data transmission can also occur. There may be technology limitations, such as limited buffer size for coordinating synchronized data transfer and consumption. Incorrect data may also result from inconsistencies in naming conventions or data codes used or inconsistent formats for input fields. Duplicate tuples also require data cleaning.

Preprocessing is required to remove noisy, missing and inconsistent data for efficient mining in Wireless Sensor Networks (WSN) data. A number of research works have been done for mining WSN data. No research work has been found to be done on preprocessing the WSN data for efficient query processing. In this project, we have evaluated a number of statistical techniques to handle missing data. Among these techniques, mean before after is found most suitable for handling missing data. We have implemented the Approximate Duplicate Record Detection method to remove the duplicate records from a dataset.

We have used some WSN datasets available in the internet for experimental purpose. K-means Algorithm has been applied for clustering the dataset. Cleaned and clustered dataset has shown better performance for query processing than dirty and non clustered data.

## Acknowledgement

My sincere thanks to my thesis supervisor, Dr. Abu Sayed Md. Latiful Hoque, for involving me in such a challenging field of cleaning and clustering of Wireless Sensor Networks (WSN) data. I am grateful to do my project under his supervision. Without his ever helping personalities, this project, would not have been completed.

I gratefully acknowledge the support and advice from Dr. Md. Humayun Kabir, Associate Professor, CSE.

My Special thanks to all teachers, students and staffs of CSE, BUET.

The completion of this project would not have been possible without encouragement of the members of my family and friends. Thank you all.

# Table of Contents

	Page No.
Abstract	i
Acknowledgement	ii
List of Tables	v
List of Figures	vi
<b>CHAPTER 1: INTRODUCTION</b>	
1.1 Background	1
1.2 Present State of the Problem	1
1.3 Objectives	3
1.4 Related Works	3
1.5 Organization of the Report	4
<b>CHAPTER 2: CLEANING AND CLUSTERING METHODOLOGIES</b>	
2.1 Data Cleaning Methods	5
2.1.1 Approximately Duplicated Records Cleaning	5
2.1.2 Key Steps of Approximately Duplicated Records Cleaning	6
2.1.3 Algorithm of Approximately Duplicate Records Detection	7
2.2 The Cleaning of Incomplete Data	8
2.2.1 Estimation of Missing Values	10
2.3 Clustering	14
2.3.1 Problems Occurred for Clustering	14
2.3.2 K-Means Clustering	15

## CHAPTER 3: RESULT AND DISCUSSION

3.1 Experimental Setup	17
3.2 Dataset Description	17
3.3 Cleaning Redundant Data	17
3.4 Estimation of Missing Values	18
3.5 Data Clustering	22

## CHAPTER 6: CONCLUSION

6.1 Conclusion	25
6.2 Future works	25
REFERENCES	26

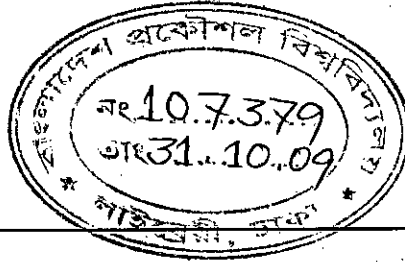
## List of Tables

<b>Table No.</b>	<b>Title</b>	<b>Page No</b>
Table 3.1:	Data Sets for Missing Values	18
Table 3.2:	Calculation for Different Statistical Methods	19



## List of Figures

<b>Figure No.</b>	<b>Title</b>	<b>Page No</b>
Figure 1.1:	Wireless Sensor Network Architecture	2
Figure 2.1:	The Flowchart of Approximately duplicate Record Detection	5
Figure 2.2:	Approximate Duplicate Record Detection	7
Figure 2.3:	The Flowchart of Incomplete Data Cleaning	8
Figure 3.1:	No. of Records Vs File Size (KB)	18
Figure 3.2:	Comparison of Prediction Accuracy for Different Methods	20
Figure 3.3:	Comparison of Coefficient of Determination for Different Methods	20
Figure 3.4:	Comparison of Mean Absolute Error for Different Methods	21
Figure 3.5:	Comparison of Root Mean Squared Error for Different Methods	21
Figure 3.6:	K-Means Applied to Dirty Data	22
Figure 3.7:	K-Means Applied to Clean Data	22
Figure 3.8:	No. of Events Vs Query Response Time	23
Figure 3.9:	No. of events Vs Query Response Time	24



### 1.1 Background

The emergence of compact, low-power, wireless communication sensors and actuators in the technology supporting the ongoing miniaturization of processing and storage, allows for entirely new kinds of embedded system. These systems are distributed and deployed in environments where they may not have been designed into a particular control path, and are often very dynamic. Collections of devices can communicate to achieve a higher level of coordinated behavior.

Wireless sensor nodes deposited in various places provide light, temperature, and activity measurements. Wireless nodes attached to circuits or appliances sense the current or control the usage. Together they form a dynamic, multi-hop, routing network connecting each node to more powerful networks and processing resources.

Wireless sensor networks are application-specific, and therefore they have to involve both software and hardware. They also use protocols that relate to both the application and to the wireless network.

Query processing is one of the challenging issues of sensor networks. For efficient query processing clean data is needed. Redundant and missing values hamper the efficient query processing. As the network is real time and dynamic in nature, efficient query processing is required for useful outputs.

### 1.2 Present State of the Problem

Data mining is required for efficiently data retrieving from data sources. Sensor nodes are limited to battery power and memory space. A suitable technique is required for mining sensor data. Dirty data is a challenge for efficient mining and hence preprocessing is required.

Smart environments represent the next evolutionary development step in building, utilities, industrial, home, shipboard, and transportation systems automation. Like any sentient organism, the smart environment relies first and foremost on sensory data from the real world. Sensory data comes from multiple sensors of different modalities in distributed locations. The smart environment needs information about its surroundings as well as about its internal workings.

The challenges in the hierarchy are detecting the relevant quantities, monitoring and collecting the data, assessing and evaluating the information, formulating meaningful user displays, and performing decision-making and alarm functions are enormous. The information needed by smart environments is provided by distributed Wireless Sensor

Networks, which are responsible for sensing as well as for the first stages of the processing hierarchy. Cleaning and clustering of sensor data is one of the challenging issue for query processing in sensor networks. For efficient mining of sensor data clean data plays an effective role. Redundant data delays the query response time. Faulty nodes are the cause of missing values. Missing values hamper our proper monitoring of output. We need to estimate missing values which can meet our requirements. Searching becomes efficient if data are clustered. It is easier to find out a particular pattern from clustered data rather than from non clustered data. Cleaned and clustered data make the query processing more efficient by improving the query response time.

Various mining techniques can be applied to the preprocessed data. Clustering is required to make the searching efficient. Searching data from a huge database is not suitable. We need high quality data for high quality mining. Sensor networks collect information from sensor nodes which may be faulty. Data transmission errors, limited memory are also challenges for clean data. Duplicate records can hamper the preprocessed tasks. A suitable technique is required for cleaning the duplicate records. After preprocessing, clustering can be applied to data. If we can cluster the data it will make the query processing more efficient.

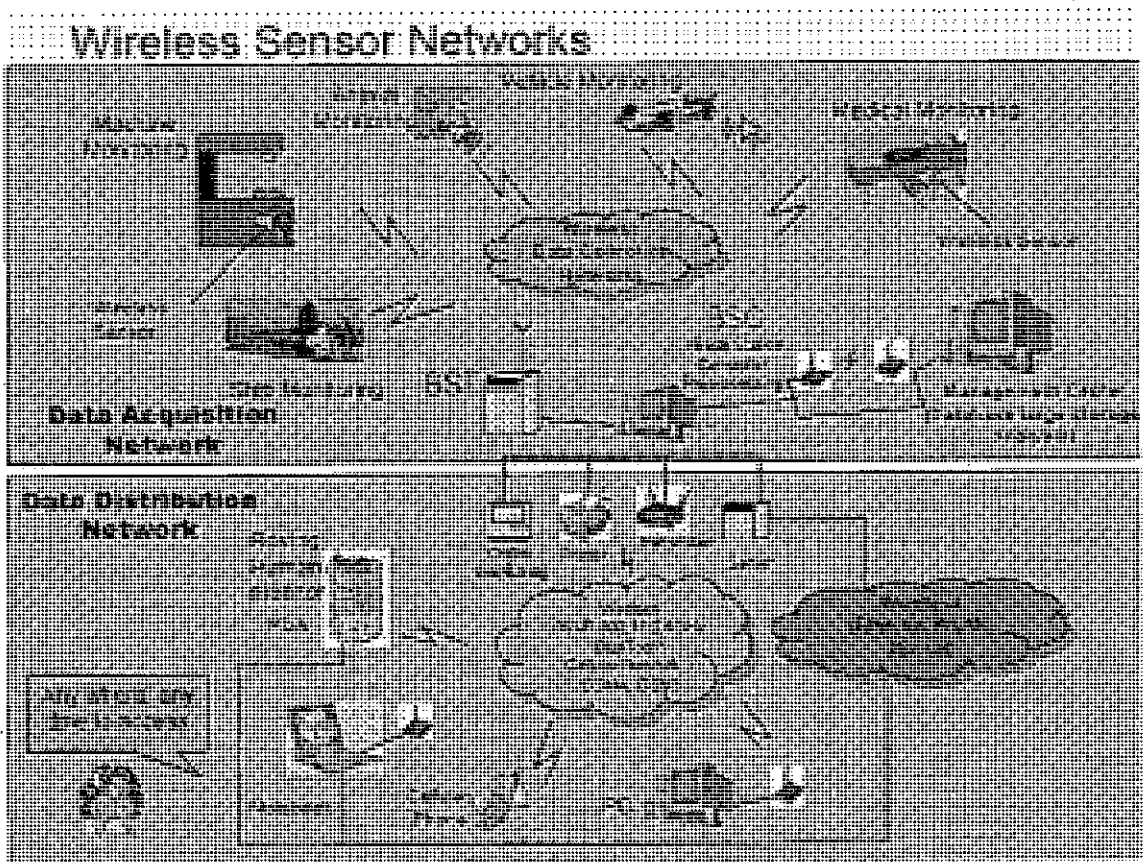


Figure 1.1: Wireless Sensor Network Architecture

### 1.3 Objectives

The objectives of this research and possible outcomes are to:

- Remove noisy, missing, inconsistent and redundant data from WSN database for efficient mining.
- Perform data clustering by K-means algorithm.
- Evaluate the performance of this data pre-processing using the representative dataset.

### 1.4 Related Works

Several works and studies have been performed regarding data mining in WSN Data. These are Artificial Neural Network based approach [1], Kohonen Map Implementation [2], Adaptive Modular Approach [3], Local Hill Climbing Approach [4], Distributed Approach for prediction [5]. Artificial Neural Network Implementation [1] presents two possible implementations of the ART and FuzzyART neural networks algorithms. It is unsupervised learning methods for categorization of the sensory inputs. They are tested on data obtained from a set of several motes, equipped with several sensors. A framework for building and deploying predictors in sensor networks that pushes most of the work out to the sensors themselves. A neural network algorithm can be implemented in the tiny platform of Smart-It units, which are kind of sensor nodes or motes. Thus instead of reporting the raw-data, each Smart-It unit can send only the cluster number where the current sensory input pattern has been classified. In that way a huge dimensionality reduction can be achieved depending on the number of sensor inputs in each unit. In the same time communication savings will benefit from the fact that the cluster number is a small binary number unlike sensory readings which can be several bytes long real numbers converted from the analog input. Kohonen Map Implementation [2] enables sensors to respond to changes in data by relearning when their local predictive accuracy changes. This creates new possibilities, such as allowing sensors to predict only some target classes. Adaptive Modular Approach [3] is a two-layer modular architecture to adaptively perform data mining tasks in large sensor networks. Lower layer performs data aggregation and upper layer performs local learning technique. data set to find the best way to combine the measures of neighboring sensors such that the accuracy of the prediction model based on such aggregate measures is optimized. The design procedure relies on an iterative optimization procedure which loops over five steps: a partition of the sensor units in proximity clusters, the compression of the signal of cluster of sensors, the aggregation and compression of signal to the upper data mining server, the training of the prediction model in data mining server, the assessment of partition according to multiple criteria. Local Hill Climbing Approach [4] shows that in some cases hill climbing can be solved using a local algorithm. Local algorithms are important for sensor networks because they have superb message pruning capabilities and because they perform their entire computation in-network. It describes a new k-facility location

algorithm suitable for sensor networks. The qualities which qualify the algorithm for this kind of systems are its message efficiency, its strong local pruning, and its ability to efficiency sustain failures and changes in the input. All these qualities stems from the algorithm's local nature. Distributed Approach [5] describes deploying predictors in sensor networks that pushes most of the work out to the sensor themselves. It shows that prediction performance is not negatively impacted by using the framework instead of a centralized learning approach.

In Artificial Neural Network Implementation, two models are presented for categorization of sensory inputs. There is no clustering and cleaning technique. In Kohonen Map Implementation, it allows sensors to predict only some target classes. It is not a proper data mining technique. In Adaptive Modular Approach, it gives a structure only. Data mining technique is not purely defined. In Local Hill Climbing Approach, it develops local algorithms for sensors which is capable of solving only the centralized problem. It solves a particular problem. Nothing is mentioned about clustering technique.

## **1.5 Organization of the Report**

Chapter 2 has covered the cleaning and clustering methodologies. Result and discussion has been discussed on chapter 3. Chapter 4 has focused on conclusion and future works.

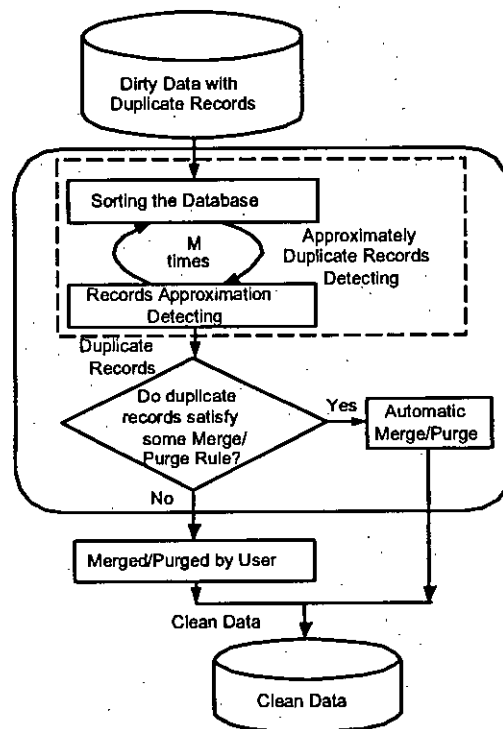
## Cleaning and Clustering Methodologies

### 2.1 Data Cleaning Method

We have applied Approximately Duplicate Record Detection method for cleaning WSN data. According to the practical needs of sensor data acquisition, this chapter mainly focuses on the approximately duplicated records cleaning and incomplete data cleaning.

#### 2.1.1 Approximately Duplicated Records Cleaning

In order to reduce the redundant information from the sensor data acquisition, it is important to clean approximately duplicated records. The approximately duplicated records refer to the same real-world entity, which cannot be confirmed by the system of database for its differences of formatting and spelling. Figure 2.1 shows the flowchart of approximately duplicated records cleaning method.



**Figure 2.1:** The Flowchart of Approximately Duplicate Record Detection

## 2.1.2 The key steps of approximately duplicated records cleaning

From Figure 2.1, the key steps of approximately duplicated records cleaning can be summarized as: sorting the database → records approximation detecting → purge/merge of approximately duplicated records, the functions of which are illustrated as follows:

### (1) Sorting the database

To locate all the duplicated records in data source, it is essential that each possible record pair be contrasted. However, the detection of approximately duplicated records becomes a costly operation. When the amount of acquired data increases enormously, this will result in an invalid and unpractical approach. To decrease the number of record contrasts, and to increase the effectiveness of detection, the general approach is to contrast the records within a limited range, i.e. to sort the database first, then to contrast the records in the neighboring range.

### (2) Records approximation detecting

It is an essential step to detect the approximation of records in approximately duplicated record cleaning. By detecting the approximation of records, we have determined whether two records are approximately duplicated records.

### (3) Approximately duplicated record purge/merge

Having completed the detection of approximately duplicated records, the detected duplicated records should be processed. For a group of approximately duplicated records, two methods have been applied:

Method 1: regard one record true in the approximately duplicated records, the rest false. The mission is, therefore, to delete the duplicated records in the database. In this situation, the following measures can be taken:

#### Manual rules

Manual rules refers to find the most accurate record to store manually from a group of approximately duplicated ones and delete all the rest from the database. This is the easiest.

#### Random rules

Random rules refers to select any one record to store randomly from a group for approximately duplicated ones and delete all the rest from the database.

#### The latest rules

In many cases, the latest records can better represent a group of approximately duplicated records. For example, the more up-to-date the information is, the more accurate it may be. The address of daily used accounts is more authorized than that of the retired accounts. So it means to choose the latest record from a group of approximately duplicated records and delete the others.

## Integrated rules

Integrated rules refers to choose the most integrated record to store from a group of approximately duplicated ones and delete the others.

Method 2: regard each individual approximately duplicated record as a portion of the whole information source, the purpose of which is to integrate a group of duplicated records to produce another more complete group of new records. This approach is generally manually done.

### 2.1.3 Algorithm of Approximately Duplicated Records Detection

**Theorem:** Given any two character strings,  $x$ ,  $y$ , the length of which are respectively  $|x|$ ,  $|y|$ . If the maximum edit distance is  $k$ , the difference between the lengths of the two character strings cannot exceed  $k$ , i.e.  $\|x - y\| \leq k$ .

The algorithm of approximation detecting algorithm is given in figure 2.2.

#### Algorithm:

**Input:** two records:  $R_1$  and  $R_2$ , the threshold of the two fields is  $\delta_1$ , the threshold of the two records is  $\delta_2$ . (the two values are to determine whether the two records are approximate)

**Output:** True/False

```
Step 1: Assign initial value for record distance Rdist = 0
Step 2: Find the field number of Record R1 and assign it to n
Step 3: For i=1 to n repeat steps 4,5 and 6
Step 4: If  $R_1$ .Field[i] == NULL OR  $R_2$ .Field[i] == NULL
        Decrease the field number by 1
        continue
Step 5: Assign s_int = length( $R_1$ .Field[i]) and t_int = length( $R_2$ .Field[i])
        If (s_int - t_int) >  $\delta_1$ 
            Return False
        Else
            Find the distance between field of R1 and R2
Step 6: If distance >  $\delta_1$ 
        Return False
        Else
            Add this distance to the record distance
Step 7: Divide record distance by record field number
Step 8: If record distance <  $\delta_2$ 
        Return True
        Else
        Return False
```

**Figure 2.2** Approximate Duplicate Record Detection Algorithm



Here, distance is used to calculate the distance between fields  $R1.Field[i]$  and  $R2.Field[i]$ .

The distance between two fields can be calculated by Euclidean distance.

$$\text{Euclidean: } \text{dis}(t_i, t_j) = \sqrt{\sum_{h=1}^k (t_{ih} - t_{jh})^2}$$

Where  $t_i$  and  $t_j$  are two tuples from  $R_1$  and  $R_2$ .

The process of approximately duplicated records cleaning can be described as follows:

Firstly, the data to be cleaned was input into system. Then, data cleaning was performed. The module of sorting the database introduces sorting arithmetic from arithmetic base in order to sort the database. Having sorted the database, the module of records approximation detecting introduces approximation detecting arithmetic from arithmetic base. Approximation detecting was done in the neighboring scope so as to 1) calculate the approximation of records, 2) to determine whether the records are approximately duplicated ones. To detect more approximately duplicated records, sorting the database once is inadequate. It is necessary to adopt multi-round sorts, multi-round contrasts, with a different key for a different round, before the integration of all the detected approximately duplicated records. In this way, the detection of approximately duplicated records was completed. Finally, the integrated disposal of approximately duplicated records was completed according to the predefined purge/merge rules for each detected group of approximately duplicated records.

## 2.2 The Cleaning of Incomplete Data

Due to being unable to obtain the value of some data attributes in collecting data, some data are incomplete. To meet the demands of auditing analysis, it is necessary to clean the incomplete data in data source. The flowchart is illustrated in Figure 2.3:

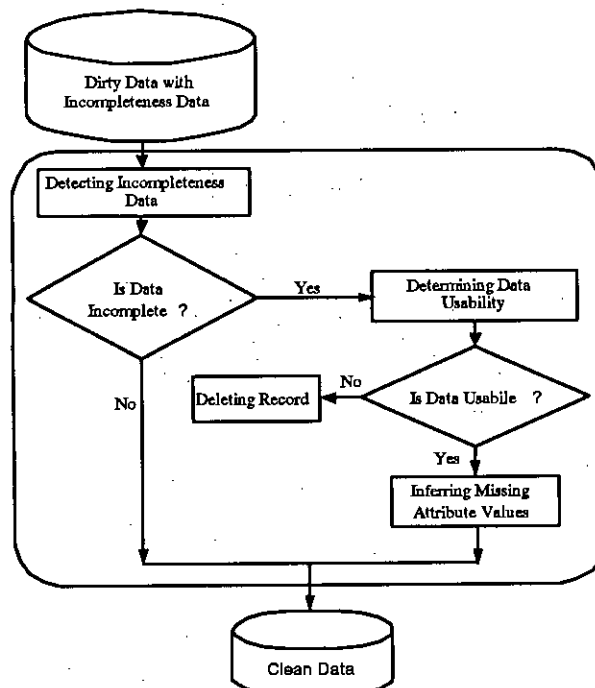


Figure 2.3: The flowchart of Incomplete Data Cleaning

The main steps of incomplete data cleaning are as follows:

(1) Detecting of incomplete data

The first step is to detect the incomplete data before cleaning so as to further cleaning.

(2) Detecting data's usability

This is an important step in incomplete data cleaning. When much value of record attributes is lost, it is unnecessary to make up all the records. In this case, it is very essential to determine the usability of records in order to seek solution to data incompleteness. To determine the usability means to decide these records should be saved or deleted based on the degree of incompleteness of each record and other factors.

The degree of incompleteness should first be evaluated. That is to calculate the percentage of lost value of attributes of any record, then to consider other factors. For instance, decide whether the key information still exists in the residuals of values of attributes, then decide whether to accept or reject. If the attributes of one record are default, this means that the values are lost. The evaluation of data incompleteness is as follows:

Let  $R = \{a_1, a_2, \dots, a_n\}$ .

Here, let  $n$  attributes of record  $R$  be denoted by  $a_1, a_2, \dots, a_n$ .  $m$  denotes the number of missing attribute values in record  $R$  (including the fields whose values are default values).  $AMR$  denotes the percentage of missing attribute values in record  $R$ ,  $\ell$  denotes the threshold of the percentage of missing attribute in record  $R$ . If:

$$AMR = \frac{m}{n} > \ell, \quad \ell \in [0,1]$$

Then, the record should be retained; else, the record should be discarded.

In cleaning incomplete data, the value of  $\ell$  is decided by the analysis of its data source by expert and is saved in the system for use. The default attribute values are also defined in the rules base for calculating the value of  $m$ .

The existence of key attributes, apart from the consideration of the incompleteness degree, has also to be taken into consideration. The key attributes are determined by the field experts according to their analysis of concrete data source. The records should also be saved even  $AMR > \ell$ , given the key attributes exist in the incomplete data.

(3) Incomplete data processing

This means to process the lost values of attributes in the records by means of some techniques after detecting the usability of data. Some methods are as follows:

**Manual method:** this is often applied for processing important data or the incomplete data when the amount is not large.

**Constant substitute method:** All missing values are filled in with the same constant, such as "Unknown" or "Miss. Value". This method is simple but may result in wrong analysis results since all missing values are filled in with the same.

**Average substitute method:** Use the average of an attribute to fill in all missing values in the same attribute.

**Regular substitute method:** The value of the attribute that occurs most often is selected to be the value for all the missing values of the attribute.

**Estimated attribute method:** this is the most complex, yet most scientific method. Use such relevant arithmetic as regress and decision tree to predict the possible value of the lost attributes and then replace defaults with predicted values. The methods given above are some usual approaches to process the lost values of attributes in record processing.

### 2.2.1 Estimation of Missing values

Data is not always available. Many tuples have no recorded values for several attributes. Missing data may occurred due to equipment malfunction, inconsistent with other recorded data and thus deleted, changes of the data. Missing data may need to be inferred.

There is a loss of information and, as a consequence, a loss of efficiency. The results may be biased due to the systematic differences between observed and unobserved data. There are several complications related to data handling, computation and analysis, due to the irregularities in data structure and the impossibility of using standard software.

We have applied various statistical imputation techniques for estimation of missing values. This section focuses on these techniques.

#### Single imputation techniques

Imputations are means of drawing from a predictive distribution of the missing values, and therefore require a method of creating such a predictive distribution based on the observed data. Complete data matrices can be created using either single imputation or multiple imputation methods. With single imputation, one value is estimated for each missing datum. It has appealing features; for example, the standard complete-data method can be applied directly, and the substantial effort required to create imputations is only needed once. Multiple-imputation is a method of generating multiple simulated values for each missing item in order to properly reflect the uncertainty attached to missing data.

We have considered six single imputation techniques. Four of these were interpolation techniques (linear, quadratic, cubic, and nearest neighbor interpolation). The remaining two were the mean imputation techniques which we will refer to as the mean-before-after and mean-before methods.

## Linear Interpolation

In linear interpolation two data points are connected with a straight line and hence the interpolation function is given by

$$f_1(x) = b_0 + b_1(x - x_0) \quad (1)$$

where  $x$  is the independent variable,  $x_i$  ( $i = 0, 1, 2, \dots$ ) is a known value of the independent variable, and  $b_i$  are unknown coefficients.

Then from (1)

$$b_0 = f(x_0) \quad (2)$$

And

$$b_1 = \frac{f(x_1) - f(x_0)}{x_1 - x_0} \quad (3)$$

In which in this case  $f = f_1$ .

## Quadratic Interpolation

If three data points are available, interpolation is carried out using a quadratic polynomial. A particularly convenient form for this estimation is

$$f_2(x) = b_0 + b_1(x - x_0) + b_2(x - x_0)(x - x_1) \quad (4)$$

The coefficients  $b_0$  and  $b_1$  are obtained from (2) and (3) with  $f = f_2$ . The  $b_2$  is obtained using-

$$b_2 = \frac{\frac{f(x_2) - f(x_1)}{x_2 - x_1} - \frac{f(x_1) - f(x_0)}{x_1 - x_0}}{x_2 - x_0} \quad (5)$$

## Polynomial Interpolation

When four data points are available, a cubic polynomial can be applied. The cubic interpolation formula has the form

$$f_3(x) = b_0 + b_1(x - x_0) + b_2(x - x_0)(x - x_1) + b_3(x - x_0)(x - x_1)(x - x_2) \quad (6)$$

The coefficients  $b_0$ ,  $b_1$  and  $b_2$  are obtained from (2-5) and  $b_3$  is obtained using-

$$b_3 = \frac{\frac{f(x_3) - f(x_2)}{x_3 - x_2} - \frac{f(x_2) - f(x_1)}{x_2 - x_1}}{x_3 - x_0} - \frac{f(x_1) - f(x_0)}{x_1 - x_0} \quad (7)$$

with  $f = f_3$

### Mean-before-after Method

Let the observed data with missing values are

$$y_1, y_2, \dots, y_{n_1}, y_1^*, y_{n_1+1}, y_{n_1+2}, \dots, y_{n_2}, y_2^*, y_{n_2+1}, y_{n_2+1}, y_{n_2+2}, \dots, y_k, y_n$$

The mean-before-after method replaces all missing values with the mean of one data before the missing value and one data after the missing value.

Thus for the above data,  $y_1^*$  will be replaced by-

$$y_1^* = \frac{y_{n_1} + y_{n_1+1}}{2}$$

and  $y_2^*$  will be replaced by the-

$$y_2^* = \frac{y_{n_2} + y_{n_2+1}}{2}$$

### Mean-before method

The mean-before method replaces all missing values with the mean of all available data before the missing values.

Let the observed data with missing values -

$$y_1, y_2, \dots, y_{n_1}, y_1^*, y_{n_1+1}, y_{n_1+2}, \dots, y_{n_2}, y_2^*, y_{n_2+1}, y_{n_2+1}, y_{n_2+2}, \dots, y_k, y_n$$

$y_1^*$  will be replaced by-

$$y_1^* = \frac{1}{n_1} \sum_{i=1}^{n_1} y_i$$

and  $y_2^*$  will be replaced by-

## Performance Indicators

$$y_2^* = \frac{1}{(n_2 - n_1 - 1)} \sum_{i=n_1+1}^{n_2} y_i$$

Four performance indicators namely prediction accuracy, coefficient of determination, mean absolute error and root mean square error are used to access the imputation methods.

Prediction accuracy (PA) is computed using

$$PA = \sum \frac{[(p_i - \bar{p})(o_i - \bar{o})]}{(N-1)\sigma_1, \sigma_0}$$

Where N is the number of imputations

$o_i$  and  $p_i$  are the observed and imputed data points respectively.

$\bar{o}$  and  $\bar{p}$  are their averages

$\sigma_1$  and  $\sigma_2$  their standard deviations.

PA values range from 0 to 1, with higher values of PA indicating a better fit.

The coefficient of determination ( $R^2$ ) explains how much of the variability in the imputed data can be explained by the fact that they are related to the observed values or how close the points are to the line. It is given by –

$$R^2 = \left[ \frac{1}{N} \frac{\sum_{i=1}^N [(p_i - \bar{p})(o_i - \bar{o})]}{\sigma_1, \sigma_0} \right]^2$$

$R^2$  takes on values between 0 and 1, with value closer to 1 implying a better fit.

The mean absolute error is the average difference between predicted and actual data values, and is given by-

$$MAE = \frac{1}{N} \sum_{i=1}^N |p_i - o_i|$$

MAE ranges from 0 to infinity and a perfect fit is obtained when MAE=0

The mean squared error is one of the most commonly used measures of success for numerical prediction. Its value is computed by –

$$RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^N [P_i - o_i]^2}$$

The smaller the RMSE value, the better the performance of the model.

## 2.3 Clustering

Clustering is similar to classification in that data are grouped. However, unlike classification, the groups are not predefined. The grouping is accomplished by finding similarities between data according to characteristics found in the actual data. The groups are called clusters.

Many definitions for clusters have been proposed:

- Set of like elements. Elements from different clusters are not alike.
- The distance between points in a cluster is less than the distance between a point in the cluster and point outside it.

### 2.3.1 Problems Occurred for Clustering

When clustering is applied to a real-world database, many interesting problems occur:

- Outlier handling is difficult. Here the elements do not naturally fall into any cluster. They can be viewed as solitary clusters.
- Dynamic data in the database implies that cluster membership may change over time.
- Interpreting the semantic meaning of each cluster may be difficult. With classification, the labeling of the classes is known ahead over time.
- There is no correct answer to a clustering problem. In fact, many answers may be found. The exact number of clusters required is not easy to determine.
- Unlike learning during a classification process, where there is some a priori knowledge concerning what the attributes of each classification should be, in clustering we have no supervised learning to aid the process. Indeed, clustering can be viewed as similar to unsupervised learning.

### 2.3.2 K-Means Clustering

We have applied K-means clustering for clustering the WSN data. It is an iterative clustering algorithm in which items are moved among sets of clusters until the desired set is reached. A high degree of similarity among clusters is obtained, while a high degree of dissimilarity among elements in different clusters is achieved simultaneously.

The cluster  $k_i = \{t_{i1}, t_{i2}, \dots, t_{im}\}$  mean of is defined as  $m_i = \frac{1}{m} \sum_{j=1}^m t_{ij}$

This algorithm assumes that the desired number of clusters,  $k$ , is an input parameter. The initial values for the means are arbitrarily assigned. These could be assigned randomly or perhaps could use the values from the first  $k$  input items themselves. The convergence criteria could be based on the squared error, but they need not be. For example, the algorithm could be stop when no (or a very small) number of tuples are assigned to different clusters. Other termination techniques have simply looked at a fixed number of iterations.

#### Algorithm:

##### Input:

$D = \{t_1, t_2, \dots, t_n\}$  (set of elements)

$K$  (number of desired clusters)

##### Output:

$K$  (set of clusters)

##### K-means algorithm:

Assign initial values for means  $m_1, m_2, \dots, m_k$ ;

repeat

    assign each item  $t_i$  to the cluster which has the closest mean;

    calculate new mean for each cluster;

until convergence criteria is met;

##### Example:

Suppose that we are given the following items to cluster:

{2, 4, 10, 12, 3, 20, 30, 11, 25}

and suppose that  $k=2$ . We initially assign the means to the first two values:  $m_1=2$  and  $m_2=4$ . Using Euclidean distance, we find that initially  $K_1=\{2,3\}$  and  $K_2=\{4, 10, 12, 20, 30, 11, 25\}$ . The value 3 is equally close to both means, so we arbitrarily choose  $K_1$ . Any desired assignment could be used in the case of ties. We then recalculate the means to get  $m_1=2.5$  and  $m_2= 16$ . We again make assignments to clusters to get  $K_1=\{2,3,4\}$  and  $K_2=\{10, 12, 20, 30, 11, 25\}$ . Continuing in this fashion, we obtain the following:



$m_1$	$m_2$	$K_1$	$K_2$
3	18	{2, 3, 4, 10}	{12, 20, 30, 11, 25}
4.75	19.6	{2, 3, 4, 10, 11, 12}	{20, 30, 25}
7	25	{2, 3, 4, 10, 11, 12}	{20, 30, 25}

Note that the clusters in the last two steps are identical. This will yield identical means, and thus the means have converged. Our answer is thus  $K_1 = \{2, 3, 4, 10, 11, 12\}$  and  $K_2 = \{20, 30, 25\}$ .

# Result and Discussion

---

### 3.1 Experimental setup

For cleaning redundant dataset we have used Jbuilder 10. For missing value estimation and other statistical calculation has been performed by SPSS 12. Clustering has also been performed by SPSS and the query response time has been observed by the simulation software OMNET++.

### 3.2 Dataset Description

The dataset that we have used for redundant data cleaning is the average daily sensed temperatures, computed from 24 hourly temperature readings. The Source is: <http://www.engr.udayton.edu>. The data fields in each file consist of month, day, year, average daily temperature (F). There are total 55,000 data values with huge redundancy.

For estimation of missing values we have used one year data of sensor readings sampled every 5 minutes embedded in the Huntington Botanical Garden. The reported tuple from each sensor pod is temperature, humidity and flux. Total data values = 65000.

For data clustering we have used the same dataset that we used for redundant data cleaning. Total data values are 55,000 and the number of cluster are four.

### 3.3 Cleaning Redundant Data

We have implemented Approximate Duplicate Record Detection Algorithm (discussed in chapter 2) in Jbuilder 10. We have used both dirty and clean data for our experiment purpose. Figure 5.1 shows the comparison between dirty data and clean data. Clean data has been obtained after applying Approximate Duplicate Record Detection. We have plotted the curve and observed the File Size in Kilo Bytes (KB) for different numbers of records. We have observed that the file size has been reduced. For example, for 16,000 records, the file size with redundant values was 711 KB and without redundant values was 680 KB. For 24,000 records, the file size with redundant values 977 KB and the file size without redundant values 920 KB. In this way we found that almost 10% data were redundant, which is very significant in case of WSN data.

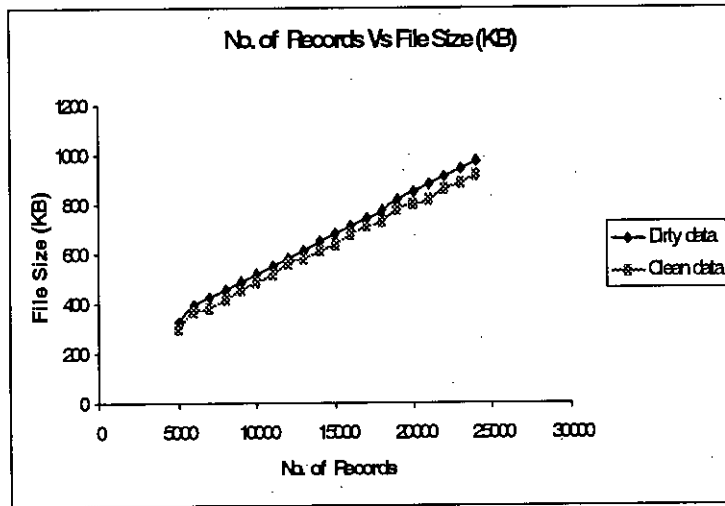


Figure 3.1: No. of Records Vs File Size (KB)

### 3.4 Estimation of missing values

We have found the number of missing values in some percentage of data sets. Table 5.1 shows the number of missing values in 5, 10, 15, 25 and 40 percentage of total data sets.

Percentage of Total Data Sets	5%	10%	15%	25%	40%
Valid	2821	5629	8418	14040	22476
Missing	429	871	1332	2210	3524
Total	3250	6500	9750	16250	26000

Table 3.1: Data Sets for Missing Values

We have used Linear Interpolation, Quadratic Interpolation, Cubic Interpolation, Nearest Neighbour, Mean-before-after and Mean-before methods for estimating missing values.

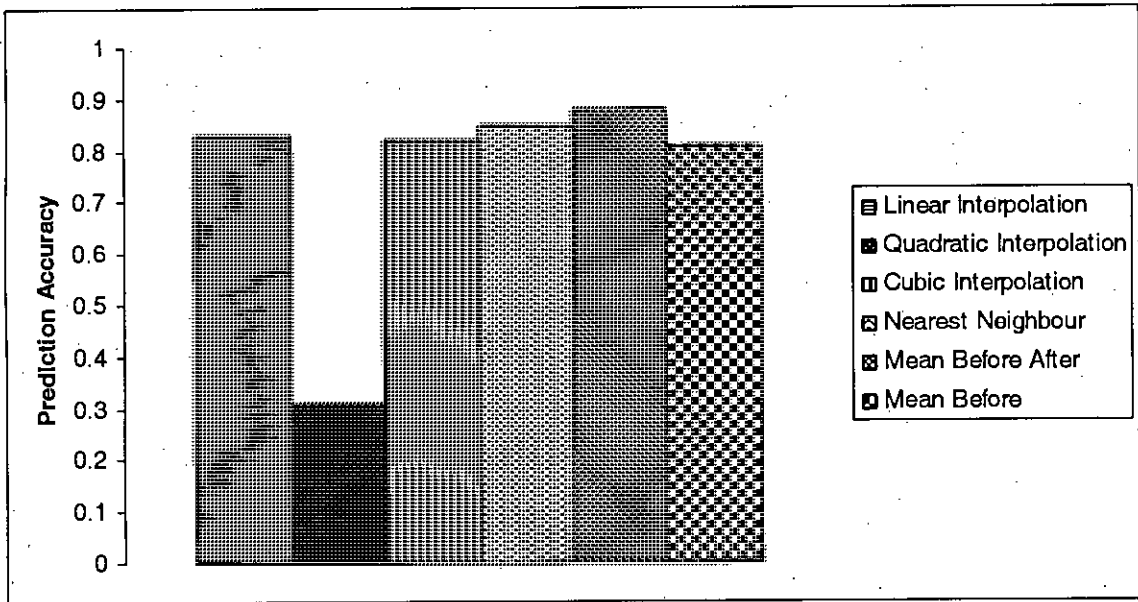
Prediction Accuracy, Coefficient of determination, Mean Absolute Error and Root Mean Squared Error have been used for performance measurement.

P=Percentage of total data sets, PA=Prediction Accuracy,  $R^2$ =Coefficient of Determination, MAE=Mean Absolute Error, RMSE=Root Mean Squared Error.

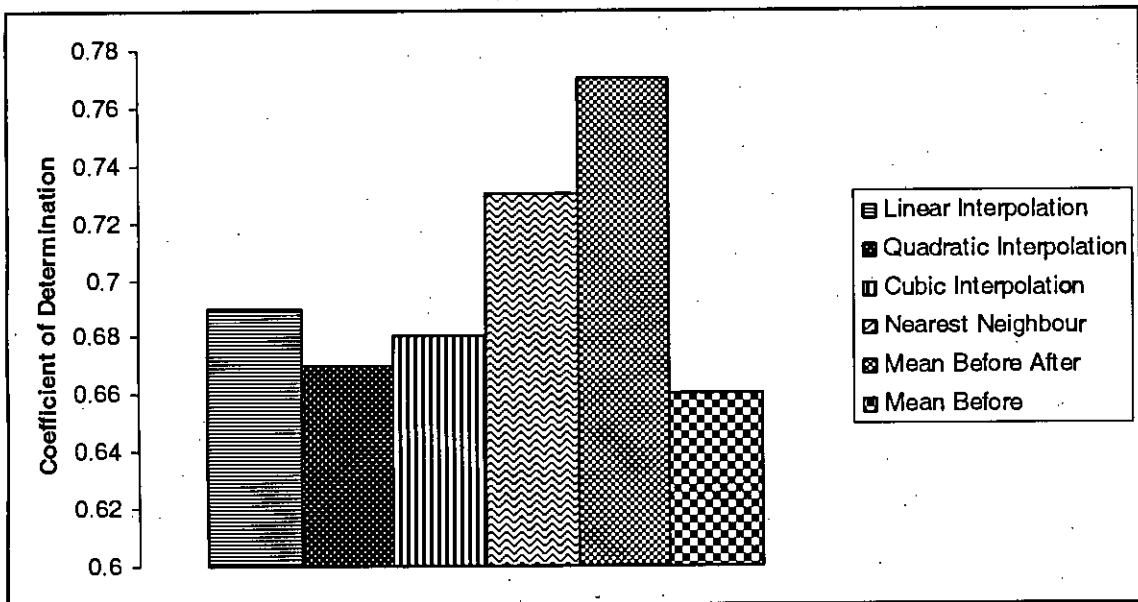
P	Method	PA	R <sup>2</sup>	MAE	RMSE
5%	Linear Interpolation	0.93	0.86	18.08	24.61
	Quadratic Interpolation	0.12	0.02	43.54	538.0
	Cubic Interpolation	0.93	0.85	18.56	25.34
	Nearest Neighbour	0.90	0.80	21.46	29.54
	<b>Mean Before After</b>	<b>0.93</b>	<b>0.87</b>	<b>17.25</b>	<b>23.71</b>
	Mean Before	0.85	0.72	25.90	35.65
10%	Linear Interpolation	0.92	0.85	17.80	25.53
	Quadratic Interpolation	0.92	0.84	18.36	26.55
	Cubic Interpolation	0.91	0.83	18.40	26.76
	Nearest Neighbour	0.90	0.80	20.77	29.84
	<b>Mean Before After</b>	<b>0.93</b>	<b>0.86</b>	<b>16.93</b>	<b>24.35</b>
	Mean Before	0.83	0.69	25.32	36.33
15%	Linear Interpolation	0.93	0.86	17.15	23.68
	Quadratic Interpolation	0.92	0.84	18.07	25.09
	Cubic Interpolation	0.92	0.85	17.55	24.21
	Nearest Neighbour	0.88	0.78	20.80	30.61
	<b>Mean Before After</b>	<b>0.93</b>	<b>0.86</b>	<b>16.61</b>	<b>23.27</b>
	Mean Before	0.84	0.70	24.11	34.78
25%	Linear Interpolation	0.89	0.77	19.21	27.82
	Quadratic Interpolation	0.87	0.76	20.19	29.44
	Cubic Interpolation	0.88	0.78	19.71	28.57
	Nearest Neighbour	0.86	0.74	21.25	31.32
	<b>Mean Before After</b>	<b>0.88</b>	<b>0.77</b>	<b>18.33</b>	<b>29.12</b>
	Mean Before	0.83	0.68	23.74	34.50
40%	Linear Interpolation	0.83	0.69	22.42	32.85
	Quadratic Interpolation	0.31	0.67	23.51	34.37
	Cubic Interpolation	0.82	0.68	23.11	34.05
	Nearest Neighbour	0.85	0.73	21.76	31.61
	<b>Mean Before After</b>	<b>0.88</b>	<b>0.77</b>	<b>19.12</b>	<b>27.79</b>
	Mean Before	0.81	0.66	24.61	35.55

**Table 3.2:** Calculation for Different Statistical Methods

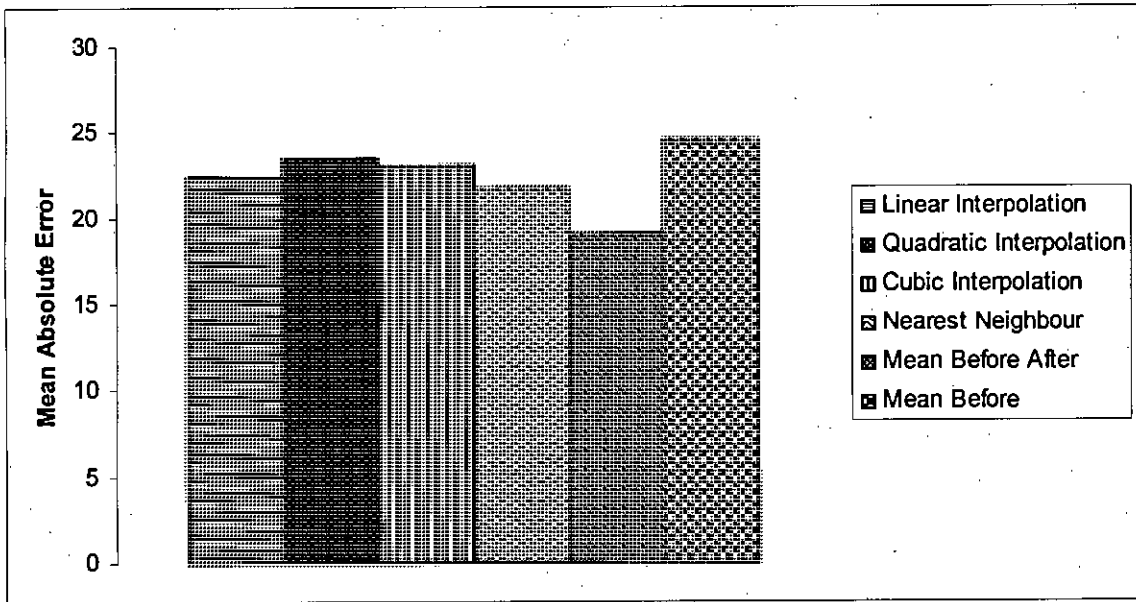
We can observe that in each percentage of data values, Mean Before After is the best method for estimation of missing values. For example, if we consider 40% values, the prediction of accuracy is 0.88 which is higher than other techniques. Higher accuracy implies better method. The coefficient of determination is 0.77 which is also higher than other techniques. The Mean Absolute Error is 19.12 which is lower than other techniques. The Root Mean Squared Error is 27.79 which is also lower than other techniques. So, Mean Before After is the best method for estimating missing values.



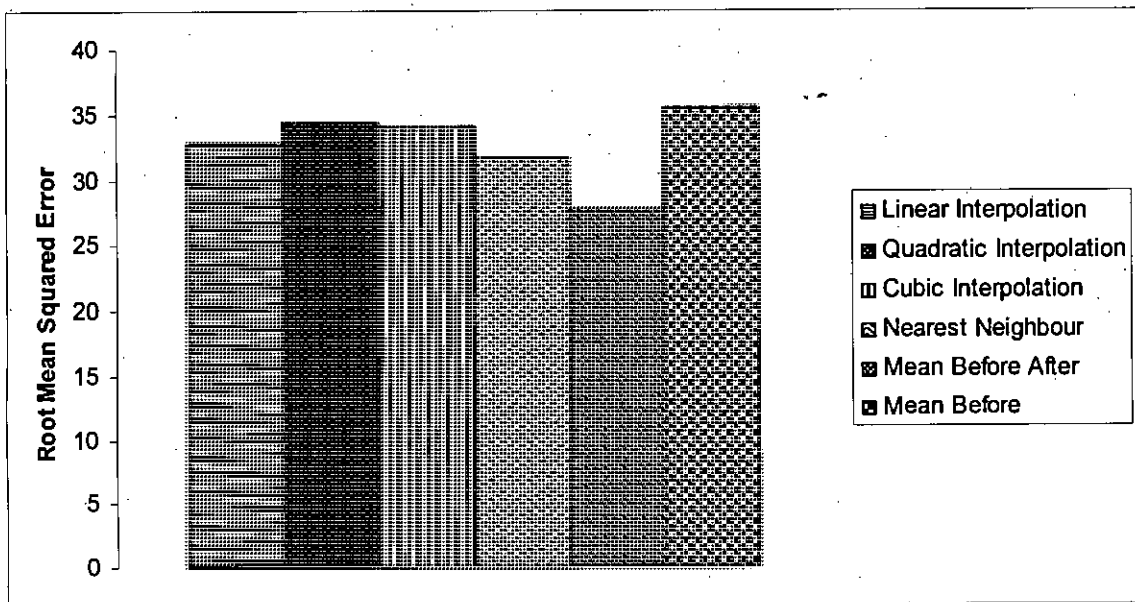
**Figure 3.2:** Comparison of Prediction of Accuracy for Different Methods



**Figure 3.3:** Comparison of Coefficient of Determination for Different methods



**Figure 3.4:** Comparison of Mean Absolute Error for Different Methods



**Figure 3.5:** Comparison of Root Mean Squared Error for Different Methods

From the Figure 3.1, 3.2, 3.3 and 3.4, we can observe that the Mean Before After is the best among six methods. Higher prediction accuracy and coefficient of determination for 40% data set shows better performance. Lower mean absolute error and root mean squared error also shows better performance than other methods.

### 3.5 Data Clustering

We have applied K-Means Clustering technique to cluster sensor data. Figure 3.5 and 3.6 show the iteration history after running K-Means clustering on datasets.

	Initial Cluster centers			
	1	2	3	4
V4	-99.00	60.30	92.50	28.10

Iteration	Change in Cluster Centers			
	1	2	3	4
1	.000	.205	11.720	11.595
2	.000	.461	3.112	3.979
3	.000	.057	.901	1.178
4	.000	.185	.163	.435
5	.000	.168	.000	.214
6	.000	.134	.041	.122
7	.000	.168	.054	.149
8	.000	.119	.054	.087
9	.000	.045	.014	.040
10	.000	.060	.000	.073

Figure 3.6: K-Means applied to Dirty Data

	Initial Cluster centers			
	1	2	3	4
V4	.00	61.80	92.50	31.10

Iteration	Change in Cluster Centers			
	1	2	3	4
1	.000	.556	11.322	10.120
2	.000	.524	3.068	3.407
3	.000	.080	.903	.949
4	.000	.003	.292	.323
5	.000	.055	.128	.073
6	.000	.018	.020	.000
7	.000	.000	.000	.000

Figure 3.6: K-Means applied to Clean Data

The K-Means clustering technique has been applied for both noisy and clean data. The simulation result has shown that the number of iterations required is less for clean data than for noisy data.

For same number of clusters and same number of data. we get better performance for clean data.

For Noisy data-

- ❖ The number of Iterations required is 10
- ❖ The minimum distance between initial clusters is 32.200

For Clean data-

- ❖ The number of Iterations required is 7
- ❖ The minimum distance between initial clusters is 30.700

Less iterations is required for clean data cause after removing dirty data, K-Means has been applied to reduced size of data. As the size of data has been reduced the minimum distance between initial clusters is also less than dirty data.

So, data cleaning can improve the effectiveness for data mining. SPSS is used as a simulator.

We have performed clustering algorithm both on noisy and clean data. Figure 5.4 shows comparison of query response time between non clustered dirty data and non clustered clean data. Figure 5.5 shows comparison of query response time between clustered dirty data and clustered clean data.

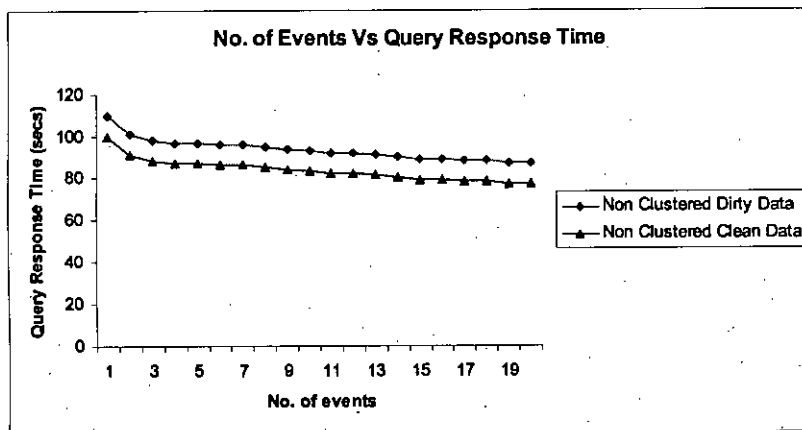
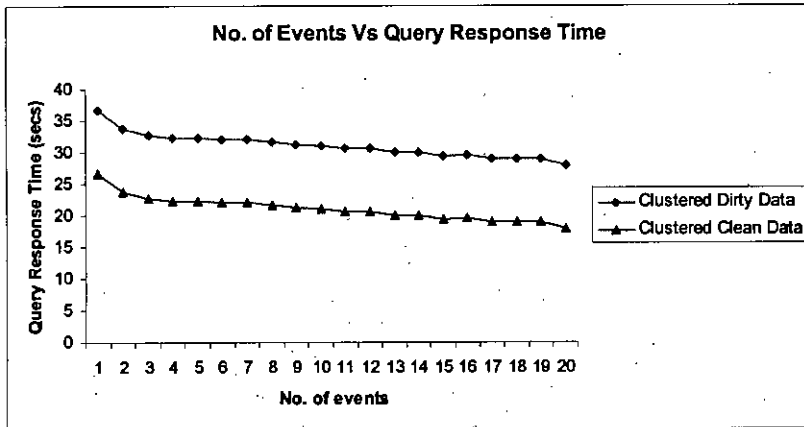


Figure 3.8: No. of Events Vs Query Response Time





**Figure 3.9: No. of Events Vs Query Response Time**

The query response time has been decreased with the increasing number of events cause we have applied both clustered and non clustered data on the WSN protocol A-sLEACH. In the A-sLEACH cluster head takes the charge of query response after some rounds and they handle the operations. So, query response time has been decreased with the increasing of events.

### **6.1 Conclusion**

Cleaning and clustering of sensor data is one of the challenging issue for query processing in sensor networks. For efficient mining of sensor data clean data plays an effective role. Redundant data delays the query response time. Faulty nodes are the cause of missing values. Missing values hamper our proper monitoring of output. We need to estimate missing values which can meet our requirements. Searching becomes efficient if data are clustered. It is easier to find out a particular pattern from clustered data rather than from non clustered data. Cleaned and clustered data make the query processing more efficient by improving the query response time. In this project, redundant data has been removed by Approximately Duplicated Records Detection method. This technique has removed almost 10% redundant values from sensor dataset which is very significant in case of wireless sensor network. Cleaned data can be clustered easily. We have used various statistical methods for estimating values. We have found from a particular dataset that in 40% of total data sets 22476 are valid and 3524 are missing. We have found Mean Before After is the best method among Linear Interpolation, Quadratic Interpolation, Cubic Interpolation, Nearest Neighbour and Mean Before methods. We have calculated Prediction Accuracy and Coefficient of Determination for different statistical methods. Mean Before After has given higher Prediction Accuracy and higher Coefficient of Determination, which has proved its efficiency. We have also calculated Mean Absolute Error and Root Mean Squared Error. Mean Before After method has shown lower errors than other methods, which has also proved its efficiency. K-means algorithm has been applied for clustering which will discover knowledge from pre-processed sensor data with less time requirement. Clustering technique has been applied both on dirty and clean data. Comparisons of query response time have been performed for clustered and non clustered data. The number of iterations both for dirty and clean data has also been compared. Better results have come for clean and clustered data. Finally, we can say that this project work has improved the query processing tasks by cleaning and clustering of WSN data.

### **6.2 Future Works**

We have used K-Means algorithm for clustering. Our future work could be apply various clustering techniques on sensor data. We have a plan to find out more efficient techniques to clean duplicate values and estimating missing values.

## References

[1] C.,Virginio, L., Luca and L., Paolo, “Challenges for data mining in distributed sensor networks”, European Commission-Joint Research Centre, IPSC, TP210 via Fermi 1,21020 Isapa, Italy. Pattern Recognition, 2006. ICPR 2006. 18th International Conference on Volume 1, Issue , 0-0 0 Page(s):1000 – 1007.

[2] K., Andrea and D., Danco, “Data mining in wireless sensor networks based on artificial neural-networks algorithms”, Computer Science Department, Faculty of Electrical Engineering, Skopje, Macedonia. 2005 SIAM International Conference on Data Mining, Sutton Place Hotel, New Port Beach, CS, April 21-23, 2005.

[3] M.,Sabine and S., David, “A distributed approach for prediction in sensor networks”, School of Computing, Queen’s University. 2005 SIAM International Conference on Data Mining, Sutton Place Hotel, New Port Beach, CS, April 21-23, 2005.

[4] B., Gianluca and B., Yann-Ael, “An adaptive modular approach to the mining WSN data”, ULB Machine Learning Group, Universite Libre de Bruxelles, Belgium. 2005 SIAM International Conference on Data Mining, Sutton Place Hotel, New Port Beach, CS, April 21-23, 2005.

[5] Computer Science Dept., Technion-Israel, “Local hill climbing in sensor networks”, 2005 SIAM International Conference on Data Mining, Sutton Place Hotel, New Port Beach, CS, April 21-23, 2005.

[6] N., Noor, S., Yahaya, A., Ramli and B.,Abdullah, “Estimation of missing values in air pollution data using single imputation techniques”, short report, ScienceAsia, page 341-345.

[7] J., Zhang, “Research on data cleaning in data acquisition”, Nanjing Audit University, Nanjing 210029, Jiangsu, China.

[8] D., Margaret, “Data Mining Introductory and Advanced Topics”, Prentice Hall Publications.

[9] For data files- <http://www.engr.udayton.edu>

