2005

An empirical study of imputation techniques for software data sets

Sumanth Yenduri
Louisiana State University and Agricultural and Mechanical College

AN EMPIRICAL STUDY OF IMPUTATION TECHNIQUES FOR SOFTWARE DATA SETS

A Dissertation

Submitted to the Graduate Faculty of the
Louisiana State University and
Agricultural and Mechanical College
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

in

The Department of Computer Science

by
Sumanth Yenduri
B.S., Andhra University, 1999
M.S., Louisiana State University, 2002
August 2005

ACKNOWLEDGEMENTS

I would like to take this opportunity to thank my **Mother** who has been most supportive and has been my source of inspiration whenever I needed direction. I would like to thank my advisor, **Dr. Iyengar** for his invaluable guidance throughout my Ph.D. studies at LSU. I learnt the importance of honesty, integrity, creativity, critical-thinking, and hard work as a researcher through him. The work proposed here is not separable from his insightful and nurturing ideas shared with me. It is with my deepest gratitude that I acknowledge him for all his help and cooperation in providing me his guidance and moral support.

I would also like to thank **Dr. Karki, Dr. Kraft, Dr. Jones, Dr. Ward** and **Dr. Gu** for being on my Ph.D. committee and giving me insightful comments regarding my research. I would also like to thank **Dr. Perkins** who guided me through my Master's Project. Finally, I would like to thank the entire faculty, the staff, and friends in the Computer Science Dept., at LSU, who have made my stay a memorable one.

TABLE OF CONTENTS

ABSTRACT

Software Project Effort/Cost/Time Estimation has been one of the hot topics of research in the current software engineering industry. Solutions for effort/cost/time estimation are in great demand. Knowledge of accurate effort/cost/time estimates early in the software project life cycle enables project managers manage and exploit resources efficiently. The constraints of cost and time can also be met. To date, most companies rely on their historical database of past project data sets to predict estimates for future projects. Like other data sets, software project data sets also suffer from numerous problems. The most important problem is they contain missing/incomplete data.

Significant amounts of missing data are frequently found in data sets utilized to build effort/cost/time prediction models in the current industry. The reasons are numerous and the missingness is inevitable. The approaches used by the companies ignore all the missing data and provide estimates based on the remaining complete information. Thus, the very estimates are prone to bias. In this dissertation, we investigate the application of a few well-known data imputation techniques (Listwise Deletion, Mean Imputation, 10 variants of Hot-Deck Imputation and Full Information Maximum Likelihood Approach) to six real-time software project data sets. Using the imputed data sets we build effort prediction models using step-wise regression analysis.

Further, we implement a hybrid methodology for imputing data by creating multiple homogenous clusters. It works in two phases. We first create homogenous clusters and then impute the missing values by selecting the appropriate donors from the created clusters. We perform useful experimental analyses and compare the impacts of these methods for enhancing prediction accuracies. We also highlight the conditions to be

considered and measures to be taken while using an imputation technique and discuss the findings and the appropriateness of each method. The main contribution is the application of clustering techniques to develop a hybrid methodology that imputes data by overcoming the limitations faced by the existing methods. The selection process of the donors implemented in our methodology is significant as it liberates the methodology from the hurdles caused by the inherent characteristics of the data set.

CHAPTER 1 INTRODUCTION

The problem of missing or incomplete data is common in many data bases [1] and is more severe in data collected through on-site surveys [2]. Little attention has been given to this problem in the field of Software Engineering [1, 7]. Significant amounts of missing or incomplete data are frequently found in data sets utilized by the effort/cost/time prediction models used in the current software industry. By knowing these estimates early in the software project life cycle, project managers can manage and exploit resources efficiently in order to meet the cost/time constraints. Traditional approaches ignore all the missing data and provide estimates based on the residual complete information. Thus, the estimates tend to be biased. To date, most companies rely on their historical database of past project data sets to predict estimates for future projects. Like other data sets, software project data sets also contain significant amounts of missing/incomplete data. Missing data create difficulty in scientific research as the statistical data analysis techniques used are not designed for them. Hence missingness causes difficulties both at the conceptual and computational levels [3].

The research in this dissertation tackles the problem of missing/incomplete data in software project data sets that are used for effort estimation. Estimates calculated from incomplete software project data sets tend to be inaccurate. Accurate estimates are needed for software project management, efficient utilization of resources and to meet constraints such as cost and timelines. Software projects are difficult to manage if we cannot possibly estimate how long and how much effort they will consume. Data imputation procedures seem to be a promising approach to fill the missing/incomplete data before making the predictions for effort/time/cost estimates. This way more

information could be used for predicting estimates. In other words by enhancing the completeness of software project data sets, better prediction accuracies on effort/time/cost estimates could be achieved.

Our goal is to analyze numerous data sets using statistical tools under various patterns of missingness, mechanisms governing missingness and data imputation. We try to show the effects of incomplete data on useful experimental analyses, how incomplete data can and probably should be dealt with, and how experiments can actually benefit from imputing data. We elaborate some potential benefits in imputing data. We intend to answer the following questions to the best of our knowledge: Does incomplete data effect predictions? When will these incomplete data models fail? and How can these prediction accuracies be improved? Are new imputation techniques a solution to encounter incomplete data?, We study the data set characteristics and missing patterns of software project data sets and would implement known as well as new methodologies on a number of real-time data sets to prove the appropriateness of data imputation in the context of software engineering data sets.

1.1 What Are Missing Values And How Are They Caused?

Missing values with in a data set arise due to lack of response or erroneous response. They include all the answers such as “no value”, “don’t know”, “unanswered”, “forgotten/skipped” so on and so forth. The reasons for missing data are numerous. To begin with, data collection is a very painstaking (in terms of both effort & time) and a costly process. The cost in collecting, reporting and maintaining data is not trivial [4, 5]. The estimates for collecting and storing data would amount from 5% -10% of the total software project cost [6]. “Wild Values” are another reason for missing values [7]. A

value is called a wild value when we know for sure that the value is not correct. For example, a categorical variable having a numerical value. Punching errors or the recorder's ignorance may be the reasons for this. The most common remedy in practice for wild values is to enter "nothing" in place of the wild value, thereby creating more missing data. Not only these, but unanswered checklists/questionnaires, skipped questions, inefficient data collection methods contribute to missingness in data sets.

1.2 The Impact Of Missing Values On Data Analysis!

Statistical methods presume that every case has information on all the variables in the data set that is being analyzed. Hence missing data reduce the statistical power [2]. Power represents the validity of the statistical inferences drawn from the data set. The inferences may represent relativity between variables, measures of dispersion or anything else. Further, estimates calculated from unreliable data sets could be biased. Currently, companies ignore all the missing information and rely on the remaining complete information in order to provide estimates. This means that the companies are using lesser information to make predictions for the future projects. Without accurate estimates, it would be a daunting task to manage software projects. Time and money wastage would be the direct consequences of inaccurate estimates.

1.3 How To Encounter The "Missing Data" Problem?

The reasons for the cause of missing data reconfirm to us that it is inevitable to have missing data. Obviously, we know the difficulties caused by missing data. Various disciplines have employed the use of "Missing Data Techniques" or "Data Imputation Algorithms" in order to reconstruct the missing data within a data set. These procedures seem to be a promising approach to counter the problem. Imputing data means filling out

the probable values for the missing data. Imputation examines the range of probable values for each variable and calculates many predicted values randomly. An analyst will end up with numerous credible data sets by using these methods. The results often produce more accurate estimates. Numerous procedures are found in the literature [3] but few software engineering researchers have employed them in their analysis. Initial research has shown that there have been better prediction accuracies when relatively simple data imputation methods were applied to the software project data sets instead of the traditional practice of ignoring missing data [1, 7, 8, 11].

Numerous reasons account to incomplete software project data sets. The most significant reason being the effort, time and cost in collecting, gathering and storing data [2, 6, 9]. Taking into account the software life cycle, it is not an easy task to collect vast amount of low-level information for each software project. Data must be collected through all stages of the software project development. Hence missing data problem would be often encountered. The impact of wild values would worsen the situation. Current remedies of ignoring such data make the software project data sets more incomplete. Such kinds of additional data loss are not desirable. The causes for missing data range from not answering the questions/checklists by the work force to punching errors. These additional data losses could affect the prediction accuracies.

The current practices of ignoring data can be devastating. One of the most commonly used techniques in the current industry is Listwise deletion [3]. This is a data ignoring method in which every case/row having one or more missing observations are deleted. There are other ignoring methods used just to decrease the amount of data being discarded such as Pairwise deletion [3]. It considers each variable separately instead of

each case/row. For every variable, all the recorded values in each observation would be used and the missing values would be ignored. If for instance, the aim is to calculate the total number of person hours for a particular variable called PERSONHR (representing the actual effort of the project in person hours), all the recorded values for the variable PERSONHR are taken into account regardless of whether they are found missing with respect to other variables in the data set. The pairwise deletion is a special case of listwise deletion. When all the variables are considered simultaneously, pairwise deletion becomes listwise deletion.

It has been proven in numerous studies that using these kinds of ignoring methods amount to huge data losses. The findings of Kim and Curry have shown that 2% of missing values (Missing at Random (MAR)) in each of the 10 variables has amounted to 18.3% of total data loss on average when using listwise deletion. 10% of missing values (MAR) in 5 variables amounted to 41% of data loss (when using listwise deletion) [10]. Such kinds of huge data losses could seriously affect prediction accuracies. Therefore use of data imputation methods to improve the completeness among software project data sets seems to work as an effective way of countering the problem of missing data. But there are other implications that are to be considered while using data imputation procedures. Although imputation can enhance the quality of the data set, choice of an appropriate imputation methodology is significant. Some methods do not preserve the relationship between variables, some underestimate variance, and some distort the underlying distributions. And hence the choice of using the proper methodology at a given instance is very significant. Diverse disciplines have been using data imputation methods to encounter the problem of missing data. But these methods have not been

utilized extensively in Software Engineering. Imputation methods particularly perform better when there is partial non-response in the data set.

There is a need to analyze the characteristics of the software project data sets by the software engineering researchers so as to devise efficient imputation strategies for the missing data problem. The software project data sets contain different kinds of variables ranging from categorical to continuous variables. It is important to study their properties. For example, a feature deletion of a categorical variable such as Project_Leader may result in more number of cases to be added to the initial data, thus boosting strategies such as Listwise deletion and Pairwise deletion.

Based upon the missingness of the values in these various variables, one would be able to predict the appropriateness of imputation for that particular data set. While using imputation strategies such as Simple Response Pattern Imputation, one needs to select a matching set of variables to select the most similar case from the data set. Studying the characteristics of data sets helps in prioritizing the matching set of variables. Percentage of missing data always plays an important role.

Another important issue is the very small sized nature of the software project data sets. Compared to the data sets in other disciplines, software project data sets tend to be small. Integration of different data sets is important but should be done with utmost care. We may encounter different problems, as companies might not be willing to share that kind of information etc. For such kinds of analysis, it is important to build a classification scheme taking into regard the very nature of software project data sets.

So far, little research has been done in exploring the implications of applying data imputation methods to software project data sets. There are a few references in the

literature related to such exploration [1, 7, 11]. All of them have quoted a significant increase in the prediction accuracy of estimates when different kinds of imputation methods were used. But all of them stress there still remains a great deal of research to be done on this topic for more concrete answers [1, 7, 11, 12, 13].

1.4 Significance and Contributions

In this study, we compare the performance of four different imputation strategies ranging from the commonly used Listwise Deletion (LD) to model based approaches such as the Full Information Maximum Likelihood (FIML) on enhancing completeness in incomplete software project data sets. We evaluate the impact of each of these methods by implementing them on six different real-time software project data sets, which are classified into different categories based on their inherent properties. The reliability of the constructed data sets using these techniques is tested by building effort prediction models using stepwise regression. Furthermore, we implement a hybrid methodology to overcome the limitations common to most traditional imputation methods. The methodology was designed by taking into aspect the missing mechanism, data set size, missing percentage and the pattern in which the data are missing. We perform useful experimental analyses and compare the impacts of the traditional approaches implemented and the proposed methodology for enhancing prediction accuracies.

The methodology works in two phases. In the first phase it creates multiple homogenous clusters. Next, it selects the donors from the clusters in order to impute missing data. A hierarchical agglomerative clustering algorithmic approach is used to form clusters which contain one or more “similar” cases. Once the clusters are formed, missing values for each case are imputed by selecting donors from that particular cluster

that they most probably would belong to. The cluster that would contribute the donor(s) is determined by calculating a proximity metric for each missing case, which determines the donating cluster. The methodology overcomes the limitations such as missing mechanism, pattern of missing data and data set size, which are often faced by the existing methods. The existing methods perform well only under few instances. The methodology is designed to perform under all kinds of scenarios. By creating homogenous clusters and selecting the most appropriate cluster for a particular incomplete case using a proximity metric makes sure that the incomplete case gets the most suitable donor(s) which is extremely important.

Moreover, the selection process of the donors implemented in the methodology is very significant as it liberates the methodology from the hurdles caused by the inherent characteristics of a data set. After selecting the appropriate donating cluster, we implement our Combination Method in order to select the most similar donor(s). We designed the Combination Method so that it works for both qualitative and quantitative variables. It takes into account the input from both kinds of variables by using two metrics for determining the donor(s), which is different from many existing methods. In fact, many existing methods work with only quantitative variables. We implemented our methodology over six real-time software project data sets and evaluated its performance with a number of existing methods. The main objectives of this study are as follows:

- To study the inherent characteristics of software project data sets such as data set size, missing mechanism, pattern of missingness etc.
- To provide a generic classification schema for all software project data sets based on their characteristics.

- To implement and test the performance of different traditional imputation methods for treating missing values in software project data sets.
- To test these methods on data sets exhibiting different characteristics and to build effort prediction models using step-wise regression in order to measure the performance of the methods implemented.
- To determine the importance of the utilization of these methods by comparison and testing, and find an efficient method for a given class of data set.
- To develop efficient hybrid methodologies and new algorithms that overcomes the limitations faced by traditional approaches.
- To test the developed methodologies on all kinds of data sets in order to evaluate their performance.
- To conduct a comparative analysis of the performance of the traditional methods with respect to the developed methodology.
- To determine and elaborate on the use of the traditional methods and the hybrid methodology by considering the different conditions and measures to be taken while using each method.
- To suggest the most efficient method that works with out bias for each kind of data set defined in the classification schema. To discuss the appropriateness of each method for imputation in the context of software data sets.

1.5 Organization Of The Dissertation

We discuss the review of literature in the next chapter. Our review focuses on usage of imputation methods in the discipline of software engineering. In the third chapter, we review the available traditional approaches for imputing data.

In the fourth chapter, we elaborate on the back ground about missing mechanisms, patterns of missing data and describe the prediction model used. Next, we describe the data sets used in the study and provide the classification schema used to classify them. In the fifth chapter, we describe the traditional methods implemented on the data sets and provide the experimental results. Later, the results are discussed.

The proposed methodology is explained in the sixth chapter. The results are noted and the performances with respect to the data sets are analyzed. In the seventh chapter, a comparative discussion on the performance of the traditional methods implemented versus the proposed methodology is laid out. In the eighth chapter, the final observations and recommendations found from the experimental analyses are discussed. The conclusions are provided in the ninth chapter. Future work and scope related to this research are noted in the same chapter. Finally, the references and vita follow.

CHAPTER 2 REVIEW OF LITERATURE

Schafer and Graham [14] said that until the seventies missing data values were handled by editing. They detailed on different methods that could be useful to impute data. The foundation work [15] on handling incomplete data was done by Rubin in 1976, which still remains the main source for imputation methods. Since then, many researchers in different disciplines employed these techniques. It was later summarized by Little and Rubin in 1987 [3] where the traditional methods were grouped into four categories listwise deletion, imputation-based procedures, weighting procedures and model-based procedures. All the basic methods were explained and laid out and this summary provides an excellent source for traditional data imputation methods.

Cox and R. Folsom [16] in the late seventies performed simulations on different missing data techniques (MDT's) and reported that hot-deck imputations performed better than listwise deletion. Though the hot-deck method gave lesser bias, they evaluated that the variance was underestimated. Ernst [17] conducted experimental studies during the same period as Cox and Folsom and agreed with the inflation of variance when hot-deck methods were used.

In 1983 [18], Kaiser showed the performance of hot-deck methods were inversely proportional to the rate of missing data in the data set. He found that with an increase in either the amount of cases containing missing values or the number of missing values per case or the combination of both them resulted in a decrease in the performance of hot-deck methods. In 1983, Browne [19] studied Listwise deletion, Pairwise deletion, Mean Imputation, and Full Information Maximum Likelihood (FIML) by Monte- Carlo simulations. He found FIML superior to the other methods. Raymond and D. Roberts in

1987 [20] showed that when missing mechanism was Missing at Random (MAR), mean imputation performed better than listwise deletion. In 1987 [21], Ford performed experiments with six different MDTs including four variants of hot-deck imputation methods and found more or less all of them performed similarly but noted that listwise deletion was outperformed by the remaining MDT's. His criteria though were estimates of mean and variance. A simulation performed by Lee and Chiu [22] showed that listwise deletion is a preferred to mean imputation when computing the polychoric correlation.

Kromrey and Hines in 1994 [23] compared the performance of five MDTs for data not missing at random. The methods included listwise deletion, pairwise deletion, mean imputation, simple regression imputation, and multiple imputation. Their findings concluded that mean imputation, simple regression imputation, and multiple imputations were inferior when missing data was not random. But, the listwise and pairwise deletion techniques performed well when the percentage of missing data was less than 30 percent. Brown in 1994 [24] compared five imputing methods listwise deletion, pairwise deletion, mean imputation, hot-deck imputation, and simple response pattern imputation (SRPI) in the context of Structural Equation Modeling to suggest SRPI was least biased.

In 1994, Roth [2] recommended using hot-deck imputation for low percentage of missing values under MCAR mechanism. Roth and Switzer [25] in 1995 performed Monte Carlo simulation to compare different MDT's. They considered listwise deletion, pairwise deletion, mean imputation, regression imputation, and hot-deck imputation under MCAR missing mechanism. They said that pairwise deletion had the least amount of dispersion and average error around true scores for bivariate correlations. When using multiple regression both listwise and pairwise deletions performed similarly. They also

recommended not using mean imputation. But Mundrom and Whitcome in 1998 [26] rated mean and hot-deck imputations performed better than regression imputation regardless of whether the variable whose values were being imputed was categorical or continuous.

Troxel, Lipsitz, and Brennan in 1997 [27] cited use of weighted procedures for data sets with nonignorable mechanism. Gelman, King, and Liu in 1998 [28] experimented multiple imputation methods on missing data. Collin et al. in 2001 [29] compared multiple imputation methods and model-based approaches (maximum likelihood) to find both performed similarly. In 2000 [30], Sebastiani and Ramoni proposed Bound and Collapse, a Bayesian framework, to assess the conditional probabilities from data sets with missing data. In 2000 [31], Gold and Bentler conducted a Monte Carlo comparison of the RBHDI (resemblance-based hot-deck imputation, which is similar to SRPI), the ISRI (iterative stochastic regression imputation), and the case-based maximum likelihood methods. The maximum likelihood performed better when the assumptions of the distribution of population are met and when there was a substantial sample size. For smaller sample sizes SRPI performed better. El Emam and others used MDT's to fill in missing values and argued hot-deck imputation performed better than simple imputation methods [11].

Kevin Strike et al. in 2001 [1] compared different MDTs for dealing with the problem of missing values in historical data sets when building software cost estimation models. Since for cost estimation models the most important performance measure is their prediction accuracy, they evaluated how this accuracy is affected by using the different missing data techniques. They investigated by using listwise deletion, mean

imputation and hot-deck imputation methods to fill the missing data. The results showed promise but the authors claim more research should be done to determine which techniques would yield minimal bias and maximum prediction accuracy. The study though was done on a single dataset taking into account only 3 productivity factors.

Ingunn Myrtveit et al. in 2001 [7] also evaluated four missing data techniques in the context of software effort modeling: listwise deletion (LD), mean imputation (MI), similar response pattern imputation (SRPI), and full information maximum likelihood (FIML). They applied the MDT's to an Enterprise Resource Project data set, and thereafter constructed regression-based prediction models using the predicted data sets. Their evaluation suggests that FIML is the appropriate imputation strategy when the data are not missing completely at random (MCAR). Unlike FIML, prediction models constructed from LD, MI and SRPI data sets will be biased unless the data are MCAR.

Shepperd et al. in 2001 [8] used Saaty's Analytic Hierarchy Process to impute missing data and estimate effort. They describe a Sparse Data Method (SDM) based upon a pairwise comparison technique and Saaty's Analytic Hierarchy Process (AHP). They showed how their approach added value to expert judgment by producing significantly more accurate and less biased results. But, validation of the techniques and choices used are based on expert judgment, which may be biased. The approach can be extended to assess a hierarchy of criteria that contribute to effort, such as function point, novelty of the task, expertise of the developers, etc. It then becomes necessary to make pairwise comparisons to assess the relative importance of each of these criteria to overall effort.

In March 2002, Colin Kirsopp and Martin Shepperd [32] elaborate a potential problem associated with empirical studies in estimating software project effort. Generally

the data set that is being analyzed is split into training and validation sets and inferences are made about the accuracies of the prediction techniques using them. Since this is done with small sized data sets, the authors state such experiments may lead to almost random results. They analyze two data sets using a configuration problem for case-based prediction and generate results from 100 training sets. This provided them to produce results with quantified confidence bounds. They conclude in both cases using less than five training data sets may lead to biased results and for an ideal survey more than 20 data sets should be analyzed.

In April 2003, Song and Shepperd [33] experimented with Multiple Imputation techniques for the problem of missing data in software project data sets. They investigated if a simple bootstrap based on a *k*-nearest neighbor method can solve the issue. They used two data sets each having cases around 20. They could not conclude if the Multiple Imputation methods were always useful for small sized software project data sets.

Naoki et al. in 2004 [34] used an effort estimation method based on Collaborative Filtering (CF) to solve the problem of missing data. The proposed method first evaluates similarity between a target (ongoing) project and each past project, using vector based similarity computation equation. Then it predicts the effort of the target project with the weighted sum of the efforts of past similar projects. However, Conventional CF cannot be directly applied to software metrics since value range of each metric is not constant while that of item ratings in CF are constant.

In May 2004, Song et al. [35] analyzed the small sized nature of the software data sets as an important characteristic and explored using simple methods of imputation for

them. They proposed a class mean imputation (CMI) method based *k*-NN hot deck imputation method to impute both continuous and categorical missing data in small data sets. To evaluate their imputation method, they used data sets with 50 and 100 observations from a larger industrial set with varying missing data percentages. They simulated by taking into consideration both MCAR (Missing Completely At Random) and MAR (Missing At Random) mechanisms. Their result suggests their new method outperformed both CMI and the *k*-NN methods when implemented individually.

CHAPTER 3 ANALYSIS OF TRADITIONAL METHODS

Table 1 depicts the various imputation strategies used by researchers from various fields. Based on the literature, the Data Imputation methods can be roughly grouped into four categories [3]: Methods Based on Complete Information, Weighting Methods, Methods Based on Imputation, Model-Based Methods. More generally, all the methods can be categorized as Random Imputation Methods and Deterministic Imputation Methods. The former methods draw imputation values randomly either from observed data or from a predicted distribution where as the latter determine only one possible value for each missing observation.

Table 1: Summary of Various Techniques Available for Data Imputation

Methods Based On Complete Information	*Listwise Deletion/Complete Case Analysis*	
	Pairwise Deletion/Available Case Analysis	
Weighting Methods	*Weighting Cell Adjustment*	
Imputation Methods	*Estimation Methods (Unconditional/ConditionalMean Imputation etc)*	
	Substitution Methods	
	Hot deck Imputation Methods	*Adjustment Cells*
		Nearest Neighbor Hood Approach Ex: k-NN Approach, SRPI
	Cold deck Imputation Methods	
	Composite Methods Ex: Regression Based Hot Deck Method etc.	
Model-Based Methods	*Regression Based Imputation Methods*	
	Stochastic Regression Imputation Methods	
	Multiple Imputation Methods	
	Maximum Likelihood Approaches such as Expectation Maximization Algorithm, Full Information Maximum Likelihood Approach	
Other Modern Methods	*Principal Components Analysis*	
	Clustering Techniques	
	Neural Networks	

3.1. Methods Based On Complete Information

All the cases with incomplete or missing information are discarded and only the complete data are used for further analysis. These are widely used strategies in the current software engineering industry. Though these methods are simple and easy to

implement, they are usually not efficient. The following are the commonly used techniques:

3.1.1 Complete Case Analysis/Listwise Deletion

In List wise deletion any case/row with one or more missing values in the data set is deleted. Only complete cases are used for further analysis. It is very simple, easy to implement and requires minimal computational time. It is the default solution in many statistical packages. It may give biased estimates and errors when the data are not missing completely in random. It generally accounts for large amounts of data losses. Lost data reduce statistical power.

3.1.2 Available Case Analysis/Pairwise Deletion

To avoid this loss, an alternative is pairwise deletion. Pairwise deletion is a special case of listwise deletion where for each variable, all the recorded values in each observation would be used and the missing values be ignored. It is also often present in most of the statistical packages. Under MCAR conditions both these methods work well. It results in the different sample sizes when each variable is considered. It generates inconsistent correlation matrix when multiple variables have missing values.

3.2. Weighting Methods

Weights are assigned to cases to make the weighted cases represent the sample of inferences as much as possible. First, a base weight is assigned that is either the inverse of the variable's selection probability or proportional to that inverse. Next, adjustments are made to compensate for the missing values. It means the base weights are adjusted according to the missing values. It is done by identifying complete cases similar to the missing cases based on some information, which is available to both. Then the weights

are changed to resemble the missing cases. Finally, they are adjusted to make the final weighted sample estimate represent the population based on a few important variables. Different kinds of weighting algorithms are used. Weighting Cell Adjustment is one such algorithm. Adjustment cells or classes are first created based on the information of the variables. Then the procedure of weighting described above takes place within each cell under a few conditions. These methods distort the distribution and inflate variance. They are implemented in a few statistical packages and they work decently under Missing at Random (MAR) conditions.

3.3. Methods Based On Imputation

All the cases that have missing observations are filled in with probable values using different kinds of algorithms and complete data sets are formed which are analyzed using different standard procedures. These perform relatively better than the previous methods.

3.3.1 Estimation Methods (Mean, Modal, Median)

They use information of other observations through different operations in order to derive plausible values for the missing case. Simplest and the most used of them is Mean Imputation. Mean Imputation (MI) works by taking into account the available observations for that particular variable and fills the missing values with the calculated mean of the available observations. It underestimates variance but is simple to implement. It is a good solution when data are both missing completely at random and normally distributed. It distorts the underlying distribution of the sample. It works for quantitative features. For qualitative features, mode calculation would be a good alternative.

3.3.2 Substitution Methods

They rely on the availability of comparable data. These methods deal missing data by filling in information inferred from other similar cases, which were previously not selected into the sample. These methods are not widely used.

3.3.3 Hot-Deck Imputation Methods

These involve filling missing values using values drawn from other complete cases (donors) in the data set. Basically hot-deck imputation selects a recorded value that best suits the missing value and replaces it. Different possible selection methods are used to replace the missing value. These methods are a common practice and can involve very elaborate schemes for selecting the donors. They reduce bias and preserve the distribution of the sample. They work well with both quantitative and qualitative variables. They choose a donor by assessing the similarity and thereby select donors, which are most similar to the imputed case. They are also available in many statistical packages. They work well under MAR conditions. Their selection criteria are simple and can be changed easily as per needs. They are computationally less demanding. But there is little theoretical work to determine their accuracy.

3.3.4 Adjustment Cells

Cells or classes may be formed for the data set and missing observations are imputed by selecting donors for each missing case from the same cell. Choice of selecting the cells is similar to weighting methods. However, these techniques may not be ideal for interval scaled (qualitative) variables as the adjustment cells are formed based on joint levels of qualitative variables and software project data sets tend to have a large number of interval scaled variables.

3.3.5 Nearest Neighbor Method

The missing values are replaced by the values of a "Nearest Neighbor" which is most similar to the incomplete case that is being imputed. The method works by finding the most similar complete cases to the incomplete case where the similarity is measured in many ways. Ex: *k*-Nearest Neighbor method, SRPI Matching Method etc.

3.3.6 Cold-Deck Imputation Methods

It is similar to the Hot-Deck methods but replace missing values of a case by a selecting a donor from an external source i.e., historical data. That is the donor comes from a previous realization of the sample. Satisfactory theoretical explanations for this method are not given [2].

3.3.7 Regression Based Imputation Methods

These methods replace missing values with predicted values based on a regression model. To begin with, a regression model is built by using all the recorded values. Each missing value is replaced by a predicted value, which is obtained by substituting the observed value for that particular value in the regression model constructed.

3.3.8 Stochastic Regression Imputation Methods

They replace missing values by a predicted value by regression imputation plus a residual, to represent uncertainty in the predicted value. Both these regression techniques are modeling techniques often used along with Hot-Deck methods.

3.3.9 Composite Methods

These methods are combined ideas from different methods. An example would be a Regression Based Hot-Deck Imputation Method. Researchers combine different methods in order to better evaluate their data samples.

3.4 Model-Based Methods

These methods assume a model for the missing data set basing inferences on the likelihood function for that model, with parameters being estimated by using approaches like the maximum likelihood. Although these methods are flexible, they are computationally demanding.

3.4.1 Multiple Imputation Methods

It is a statistical procedure in which each missing value in the incomplete data set is replaced by a set (m) of probable values, thus generating m complete data sets for further analysis. Typically m takes value between 3-10. It assumes data are missing at random. It increases efficiency of estimation. But it is computationally demanding. It does not distort the underlying distribution nor does it inflate variance. It has been implemented in a few statistical packages. These methods can be used with any kind of data and any kind of model, and they allow use of variety of statistical techniques for analysis. They produce consistent and asymptotically efficient estimates under MAR conditions. They may also be well adapted to Non-Ignorable conditions. But a big disadvantage is these methods are difficult to implement. They produce different estimates every time they are used as randomization is introduced in the imputation step.

3.4.2 Maximum Likelihood Approaches

A method of estimating parameters of a given sample by the value that maximizes the likelihood of a sample. Maximum likelihood estimation begins with an expression known as a likelihood function for the sample data. The likelihood of a sample of data is the probability of obtaining that particular sample of data given the chosen probability model. This expression contains the unknown parameters. Those values of the parameters

that maximize the sample likelihood are known as the maximum likelihood estimates. They are implemented in a couple of statistical packages. These work well under MAR conditions. Some researchers have proven that these techniques perform decently under Non-Ignorable conditions as well. These techniques are computationally demanding though. They do not serve for general purpose and often researchers encounter practical complications with these procedures.

3.4.3 Expectation Maximization (EM) Algorithm

This is a general method for obtaining the maximum likelihood estimates of parameters in problems with incomplete data. It can be summarized in four steps:

1. Replace missing values by estimated values.
2. Estimate parameters of the distribution for the variables.
3. Re-estimate missing values assuming that the new parameter estimates are true.
4. Re-estimate parameters in an iterative procedure until convergence.

It converges reliably but the rate of convergence can be very slow if there is a huge amount of data missing. There are many more approaches such as Full Information Maximum Likelihood Approach etc.

3.5 Other Modern Methods

3.5.1 Neural Networks

Neural networks are now being employed as imputation tools. The algorithms provide non-linear input-to-output mappings. A neural network is constructed and synaptic weight matrix is trained using the complete cases of the data set. The missing values are then estimated for each input using the trained weight matrix. The synaptic weight matrix is again trained using the generated completed data set.

3.5.2 Principal Component Analysis

Principal component analysis (PCA) involves mathematical procedures that reduce the number of variables and detect structure in the relationships between variables. In the context of missing values PCA can be used in combination with other methods like the EM algorithm. Principal components analysis for missing data is being used along with a wide variety of algorithms in recent times.

3.5.3 Cluster Techniques

These methods uncover the structure of the sample data set. They divide the data set into smaller groups and repeat the process until the cases within each group are as similar as possible. By creating homogenous clusters of similar cases, they help researchers in designing selection processes that pick the most suitable donors for an incomplete case.

3.6 Tools/Software/Statistical Packages

Table 2: Statistical Packages/Tools/Software Available for Various MDT's

Tools/Software	Features	Other Details
SAS Base	Mean Imputation	Not Free, Easy to Use
SAS/IML	Multiple Imputation	Free, but difficult for beginners
SAS EM_COVAR.SAS	Expectation Maximization Algorithm	Free, but difficult for beginners
SPSS Base	Mean Imputation	Not Free, Easy to Use
SPSS Missing Values Analysis (MVA)	Listwise/Pairwise Deletion, Regression Methods, Expectation Maximization Algorithm	Not Free, Easy to Use
AMOS	Full Information Maximum Likelihood	Free, Easy to Use
MX	Full Information Maximum Likelihood	Free, but difficult for beginners
S-Plus (NORM, CAT, MIX, & PAN (4 different software available for different MI methods))	Multiple Imputation	Free and Moderate Level of Difficulty

(table 2 cont.)

SOLAS	Mean Imputation, Hot Deck Methods, Regression Methods, Multiple Imputation	Not Free, Easy to use
Prelis & Lisrel	(FIML) Full Information Maximum Likelihood	Free, Easy to use
GENESIS	Mean Imputation, Hot Deck Methods, Regression Methods	Not Free, Easy to use
SEVANI	Hot Deck Methods, Regression Methods	Not Free, Easy to use
IVEWARE	Single & Multiple Imputation	Free & Flexible
EMCOV	Expectation Maximization Algorithm	Free, Moderately Difficult to use
MISTRESS	Multiple Imputation	Free, Easy to use
MPlus	FIML	Free, Easy use

CHAPTER 4 PRELIMINARIES AND TERMINOLOGY

In this chapter, we discuss the preliminaries, models and other terminology used in the study. We first discuss the different kinds of missing mechanisms exhibited by data sets. Next we explain the three different ways in which data can be found physically missing. Further, we explain the method used in building an effort prediction model, which in our case is the step-wise regression analysis. Finally, we elaborate our classification schema using which any software project data set can be categorized.

4.1 Ignorable and Non-Ignorable Missing Mechanisms

Handling missing data is dependent upon the how the data are missing. It is imperative to methodically categorize the data. Missing data mechanisms are classified by Rubin [3] as Ignorable and Non-Ignorable (NI). Often researchers assume that the missingness is Ignorable. Furthermore, Ignorable missing data mechanism is classified into Missing Completely at Random (MCAR) and Missing at Random (MAR).

4.1.1 Ignorable Missing Data Mechanisms (MAR, MCAR)

The data are *Missing at Random* (MAR) means that the probability that the observations are missing may be dependent on Y_o but not on Y_m. (where Y represents our data set in matrix form. Y_o represents the observed values in Y and Y_m represents the missing values in Y.)

$$P(Y|\ Y_m, \delta) = P(Y|\ Y_o, \delta),$$

conditional on a set of predictor variables δ. The MAR mechanism means the probability of Y being missing may be dependent on Y_o but not Y_m. It means that missingness is not related to the missing values but may be related to the observed values of other variables in the data set. Cases with incomplete data differ from cases with complete data, but the

missing pattern is predictable from other variables rather than being due to the specific variable on which the data are missing. For example, incompetent programmers may not want to answer all the questions on the productivity factor documents in order to hide their performance. The reason for missing data under MAR conditions is because of an external effect. MAR depends on the data and the model [36].

The data are *Missing Completely at Random* (MCAR) means that the probability that the observations are missing is not related to either Y_o or Y_m.

$$P(Y|\ Y_m) = P(Y|\ Y_o),$$

It means the missingness is not dependent upon the values of any of the other variables in the data set (missing or observed). Cases with complete data are indifferent from cases with incomplete data. For example, suppose a personnel shuffles unadjusted productivity factor documents and arbitrarily discards some of them. If the observed values were a random sample of the complete data set, complete case analysis would give the same result similar to that of a complete data set.

This is special case of MAR. It is more restricted. This mechanism is very easy to deal with but unfortunately data are seldom MCAR. This situation arises because the data were missing by design. The data can be tested for this condition (SYSTAT and SPSS MVA have implemented this feature) [3]. No such tests are available for MAR condition. If the parameters of the data model and the missing parameters are different, then the missing data mechanism is Ignorable.

4.1.2 Non-Ignorable Missing Data Mechanism (NI)

Nonignorable (NI) means the probability that the observations are missing may be dependent on Y_m but not on Y_o. Missingness is related to Y_m, it is non-random and it

cannot be predicted from other variables of the data set. This situation arises because the missing pattern can be explained but it can be only explained by the variables where data are missing. For instance, if the personnel responsible for answering the questionnaires using online forms are more likely to fill in information about their productivity factors. And suppose we cannot predict which personnel use online forms from the other variables. Under such conditions, the missing mechanism is Non-Ignorable. This is the most difficult condition to deal with.

Ignorability is a judgment made by the data analyst and it depends both on the missing data mechanism as well as the data. In practice it is usually difficult to meet the MCAR assumption. MAR is an assumption that is more often used. Schafer and Graham [14] state: "*When missingness is beyond the researcher's control, its distribution is unknown and MAR is only an assumption. In general, there is no way to test whether MAR holds in a data set, except by obtaining follow-up data from nonrespondents or by imposing an unverifiable model.*" Rubin [15] suggested that when dealing with real data, data analyst should explicitly consider the process that causes missing data. For example, we might look at survey sampling containing missing data, where only a few variables are observed for all units in the population and a few survey variables are "missing" for units that are not given importance. The mechanism causing missing data would then be the process of variable collection. If variables are given importance in such a way, the mechanism is under the control of the data analyst and may be assumed "ignorable" [2].

4.2 Patterns of Missing Data

Let X_1 to X_k be the variables represented in a matrix form. If all the values are observed and if X_k has i values completely observed, then we say that the data are missing

in univariate pattern (Fig 1). If X_1 to X_k are ordered in such a way that if Xj is missing for a unit, then X_{j+1}.... X_k are missing too (for that particular unit). Such a pattern is called monotonous pattern (Fig 2). Finally if the values are missing in a haphazard fashion in which any variable may be missing for any unit, then we say that the data are missing in arbitrary pattern (Fig 3) [3].

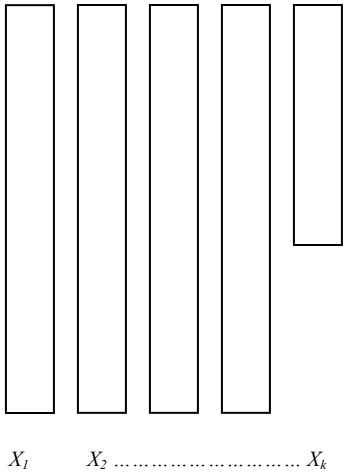


Fig 1: Univariate Pattern

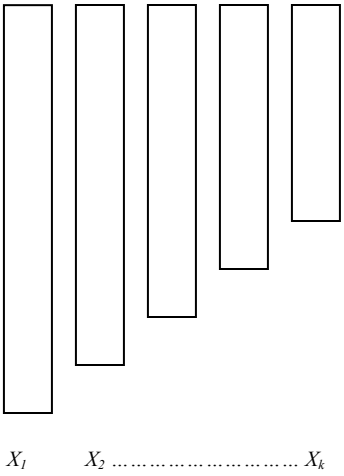


Fig 2: Monotone Pattern

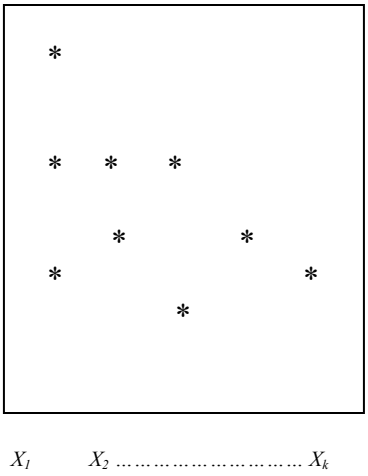


Fig 3: Arbitrary Pattern

4.3 Stepwise Regression Model

To study the impacts of these methods, the imputed data sets were evaluated using prediction models. A significant step in the construction of a prediction model is the selection of independent variables. We used the Forward Entry Stepwise Regression Model-Building Procedure.

To begin with, an initial model is identified. It always includes the regression intercept. Next "iterative stepping" is performed. That is changing the model repetitively by adding or removing a predictor/independent variable, which is based on the "stepping

constraints (tests)". Finally the termination procedure is initiated when stepping cannot be done any more or if the maximum number of steps has been reached [51, 52, 53, 54].

Initially, among all the independent variables, one variable is selected to enter the model. The independent variable that minimizes the residual sum of squared deviations and has a regression coefficient significantly different from zero is selected. Let X_1, X_2, ... X_p be the independent variables and β_1, $\beta_2 ... \beta_p$ be the regression coefficients associated with the variables respectively (Y is the dependent variable). Then the hypothesis H: $\beta_i = 0$ is rejected in order to enter the variable X_i into the model.

After the selection of the first variable, we select the second variable X_j from the remaining set such that the residual sum of squared deviations for the second selected variable combined with that of X_i is minimum and the partial correlation coefficient β_j of the second variable is significantly different from zero. The hypothesis H: $\beta_j = 0$ is rejected in order to enter the variable X_j into the model. Once X_j is entered, a test is performed to see if the first variable X_i should be included given that X_j is present in the model. If H: $\beta_i = 0$ is rejected both the variables remain or else X_i is removed. Thus the iterative process continues until the stepping criterion fails or if the maximum number of steps is reached [55, 56, 57, 58, 59].

4.4 Dataset Description

We acquired six real-time software project data sets in the past one year period from six different companies nationally and internationally. We obtained three small sized software project data sets, two medium sized and one large sized data set. We implemented different kinds of imputation methods on these six data sets. We also implemented our methodology on these data sets in order to compare and evaluate its

performance. Fig 4 shows the percentage of missing data in each of the data sets and fig 5 shows the number of variables having missing data in each of the data sets used in the experimental analysis. Details about the characteristics of each of the data set are explained in fig 6 below. It explains the numerous characteristics of each of the data sets such as missing mechanism, size, physical missing pattern, percentage of missing data etc.

Fig 4: Percentage of missing data in each data set

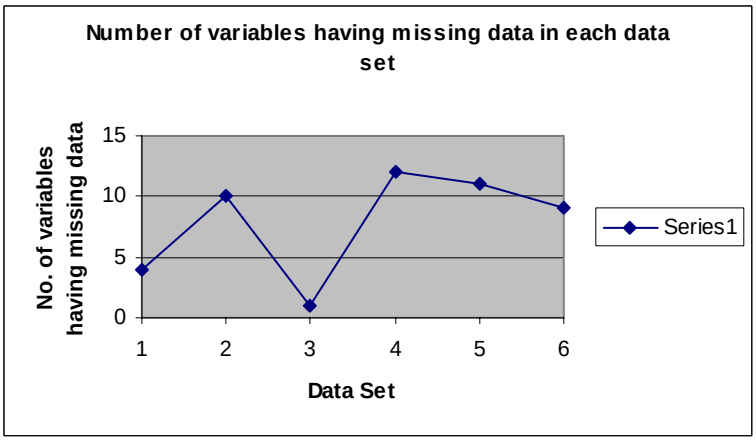


Fig 5: No. of variables having missing data in each data set

Data Set	Size	Project Type	Completion Time (years)	Missing Mechanism	% of missing data	Missing Pattern	No. of variables	No. of Cases	No. of Categorical Variables	No. of Continuous Variables	No. of Variables having missing values	Values on Dependent Variable (Y)
D1	S	Medical	5	MAR	12	A	9	21	4	5	4	NM
D2	S	Customer Service	4	MAR	32	M	12	29	3	9	10	NM
D3	S	Web Focus	2	MCAR	4	U	8	17	4	4	1	NM
D4	M	Bank	6	MAR	26	A	22	42	10	12	12	NM
D5	M	Customer Service	9	MAR	46	A	15	67	6	9	11	M
D6	L	Network Management	10	NI	18	A	23	103	8	15	9	NM

Size (S-small, M-Medium, L-Large)
Missing Pattern (U- Univariate, M – Monotonous, A – Arbitrary)
% of missing data is rounded values
Values on Dependent Variable (Y - Effort Expended for Completing the Project in Person hours) (M- Missing, NM – Not Missing)

Fig 6: Description about the Real-Time Dataset Used in the Study

4.5 Classification Scheme

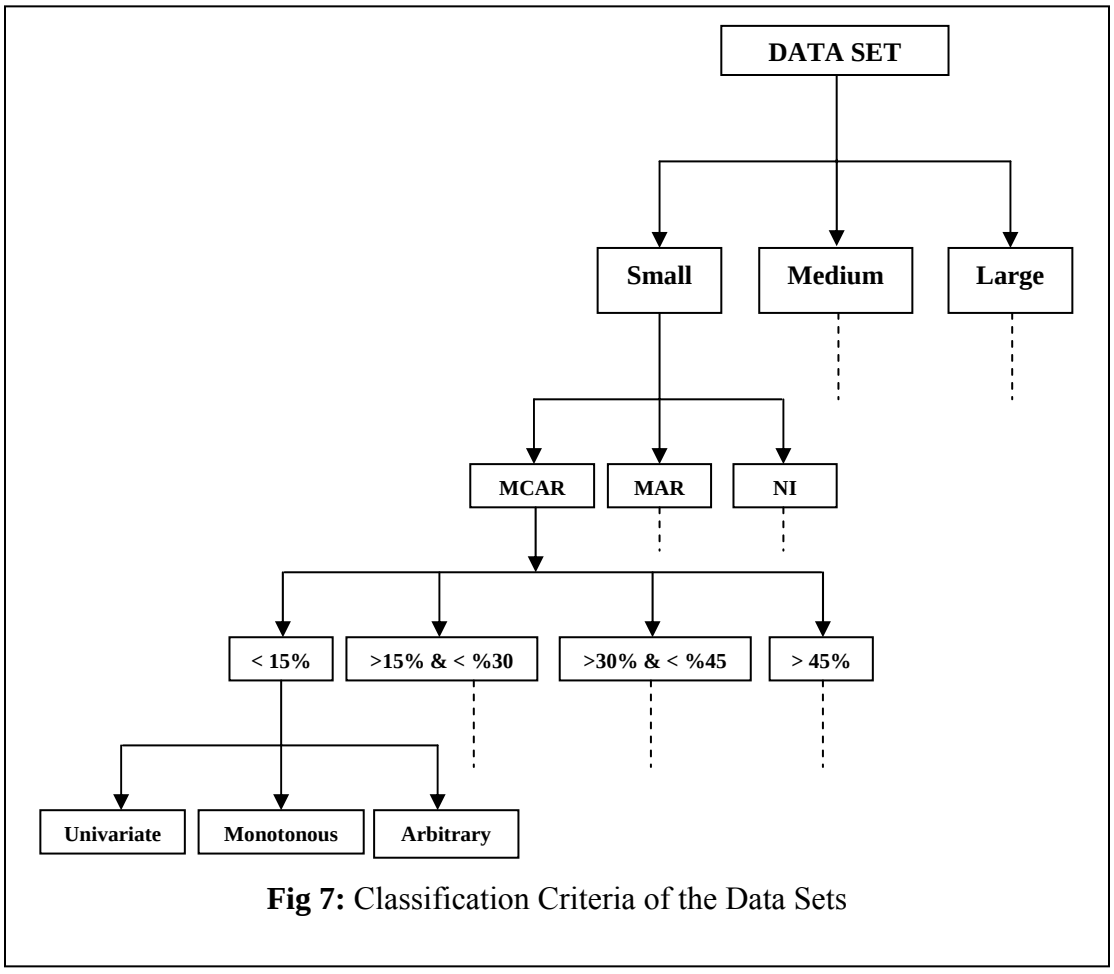


Fig 7: Classification Criteria of the Data Sets

We have classified the software project data sets based on missing mechanisms and the characteristics unique to them. Using our classification scheme, each data set can be classified and by using this classification, appropriate imputation strategy can be selected.

We classify software project data sets based on 4 parameters, namely, the size of the data set, the missing mechanism of the data, the percentage of missing data and finally the missing pattern of the data. The classification process proceeds in the same order. That is first a data set's size is determined. The attributes for size are small, medium and large. Here small indicates data set representing less than 30 cases, medium represents greater than 30 but less than 100 cases and large indicates greater than or equal to 100 cases. Each data set is classified as a small/medium/large sized data set. Software project data sets are generally small or medium sized.

The next step involves determining the mechanism in which the data are missing within the data set. The data set is then sub-classified based on whether the missing mechanism is Ignorable or Non-Ignorable. The missingness mechanism is often assumed to be Ignorable but some times it may be the other way too. Next, the percentage of missing data is determined. The data set is selected into one of the 4 subclasses here. That is < 15% of missing data, > 15% & < 30% of missing data, >30% & <45% of missing data and >45% of missing data. Finally, they are sub classified based upon the pattern of missing data i.e., univariate, monotonous or arbitrary. The missing pattern is more often arbitrary in software project data sets. The classification levels are depicted in Fig 7.

CHAPTER 5 MODELS PROPOSED AND IMPLEMENTED

In this chapter, we explain all the different methods implemented on the six real-time data sets in our study. We implemented Listwise Deletion (LD), Mean Imputation (MI), 10 variants of Hot-Deck Imputation and Full Information Maximum Likelihood Approach. We also detail on the metrics used to evaluate the performance of the methods and finally discuss the results.

5.1 Listwise Deletion

In List wise deletion (LD) any case/row with one or more missing values in the data set is deleted. Only complete cases are used for further analysis. It is very simple, easy to implement and requires minimal computational time. It is the default solution in many statistical packages. It may give biased estimates and errors when the data are not missing completely in random. In addition, it can reduce variance, inflate adjusted R^2 value, and decrease the efficiency since standard errors and t-tests are a function of sample size. We implemented LD in order to compare its performance with other methods (as it is the default method used in the current industry).

5.2 Mean Imputation

Mean Imputation (MI) works by taking into account the available observations for that particular variable and fills the missing values with the calculated mean of the available observations. It underestimates variance and inflates the adjusted R^2 value but the technique is simple to implement. It is a good solution when data is both missing completely at random and normally distributed. It works for quantitative features. For qualitative features, mode calculation would be a good alternative. We implemented MI as studies suggest that even simple methods can improve prediction accuracies.

5.3 Hot-Deck Methods

It involves filling missing value with another value drawn from other complete cases (donors) in the data set. Basically hot-deck imputation selects a recorded value that best suits the missing value and replaces it. Different possible selection methods are used to replace the missing value. These methods are a common practice and can involve very elaborate schemes for selecting the donors. They reduce bias and preserve the distribution of the sample. They work well with both quantitative and qualitative variables. They choose a donor by assessing the similarity and thereby select donors, which are most similar to the imputed case. They are also available in many statistical packages. They work well under MAR conditions. Their selection criteria are simple and can be changed easily as per needs. They are computationally less demanding. They have been implemented because of their vast usage and proven performance.

5.3.1. Sequential Hot-Decking

The procedure starts sequentially from the beginning (the first case) of the data set. The closest preceding complete case was used as a donor to impute the missing values in each incomplete case. If the first case was incomplete, a set of initial values determined from a cold deck was used to impute it. Imputed cases were not used as donors. However, the first case was used as a donor even if it were imputed. Time wise, the cases would be closer. But, there is one possible disadvantage with this method, if a number of incomplete cases occur simultaneously, the values of a same donor would be imputed in all the cases. To overcome this problem, Random Hot-Decking was used.

5.3.2. Random Hot-Decking

Here for each missing case, a donor was randomly selected from the complete set.

5.3.3. Simple Response Pattern Imputation (SRPI)

First, a matching set of variables was determined, represented by M. M is determined by careful analysis of the data set and is generated only by the analyst. For each incomplete case, all cases with values complete with respect to the missing values in the incomplete case were considered donors. The similarity between the incomplete case and each of the donors was measured using a distance metric given by the Euclidean distance. Of course, each donor had to be complete with respect to the matching variables as the Euclidean distance metric was calculated by using only the variables given by M. The complete case that gives the smallest value for the distance would act as the donor. In an event where more than one donor satisfied all the conditions, one of them was picked randomly. The method was implemented as described in [7].

5.3.4 *k*- Nearest Neighbor Method

This technique uses other complete cases (as donors) within the data set to impute an incomplete case. The missing values are replaced by the values of a "Nearest Neighbor" which is similar to the incomplete case. The method works by finding "*k*" most similar/nearest complete cases to the incomplete case where the similarity is measured by a distance parameter. The value of "*k*" was set to 2. That is, the 2 most similar/nearest cases were selected to impute the values in the incomplete case.

All qualitative variables were dummy coded so that they could be used in calculating the distance metric. Seven different distance metrics were used to form seven different complete data sets. SRPI is closely related to the *k*-NN approach where k=1, differing only in the way the similarity is computed. The method was implemented as described in [1, 37]:

The data set was divided into two sets, the cases with incomplete or missing values (Incomplete Set) and the complete cases (Complete Set). Let x_i be the vector of all the variables measured for the i^{th} case in the incomplete set and x_{ij} would be the value for the j^{th} variable measured on i^{th} case. Similarly, y_k be the vector for all the variables measured for the k^{th} case in the complete set, and y_{kj} be the value for the j^{th} variable measured on k^{th} case. The following distance parameters were calculated to different complete data sets [1]:

5.3.4.1. Euclidean Distance

It measures the distance between two points represented by a n by p matrix. In our case n is the number of cases and p is the number of variables in our data set. The metric is derived from Minkowski distance function.

$$\text{Euclidean}_{\text{ki}}\,(\text{d}) = \sqrt{\sum_{j=1}^{n}(y_{kj} - x_{ij})^2}$$

5.3.4.2. Manhattan Distance

It is the sum of the absolute differences between two points.

$$\text{Manhattan}_{\text{ki}}\,(\text{d}) = \sum_{j=1}^{n}\left|y_{kj} - x_{ij}\right|$$

5.3.4.3. Maholanobis Distance

It is the normalized distance between two N dimensional points in a multidimensional space, defined by the correlations of the independent variables. If the variables are not correlated, it becomes Euclidean distance. In other words, Maholanobis distance becomes Euclidean distance if the covariance matrix is an identity matrix. This measure tells if or not an observation is an outlier with respect to the values of the independent variables. It is given by:

$$\text{Maholanobis}_{ki}\ (d^2) = (y_k - x_i)\ C^{-1}\ (y_k - x_i)'$$

Where *i* is the missing case, *k* is the complete case and *C* is the covariance matrix.

5.3.4.4. Correlation Distance

The correlation coefficient (*r*) is a measure of linear relationships between two samples/vectors. "*r*" is given by

$$r = \frac{n\sum_{j=1}^{n} y_{kj} x_{ij} - \left(\sum_{j=1}^{n} y_{kj}\right)\left(\sum_{j=1}^{n} x_{ij}\right)}{\sqrt{\left[n\sum_{j=1}^{n} y_{kj}^2 - \left(\sum_{j=1}^{n} y_{kj}\right)^2\right]\left[n\sum_{j=1}^{n} x_{ij}^2 - \left(\sum_{j=1}^{n} x_{ij}\right)^2\right]}}$$

Similarity (S) between to vectors can be measured by, (S) = (r+1)/2.The value of similarity is between 0 and 1. Similarity increases as it approaches 1. The distance between them is given by D = (1-r) (One minus the sample correlation between the two vectors). Similarity increases as the distance between them decreases. *r* ranges from 1 for equivalent vectors to -1 for opposing vectors, so the distance metric varies from 0 to 2.

5.3.4.5. Cosine Distance

CS_{ki} is given by

$$CS_{ki} = \frac{\sum_{j=1}^{n} y_{kj} x_{ij}}{\sqrt{\sum_{j=1}^{n} y_{kj}^{\ 2} \sum_{j=1}^{n} x_{ij}^{\ 2}}}$$

The cosine similarity function between two vectors CS_{ki} [38] (Ochini Coefficient) measures the cosine of the angle in between them. That is their dot product divided by their magnitudes. One minus the cosine of the angle between them gives us the distance. The similarity is measured by cosine of the angle and distance is measured by arccosine (1- Cosine of the angle) of that value.

5.3.4.6. Squared Chord Distance

For Squared Chord distance metric, it may be necessary to have non-negative values in the data set. It is noted that the values be shifted to non-negative (or positive) values before calculating these distances. Also other metrics such as Canberra and Squared Chi-squared distances could be used but again the values need to be shifted or else infinite or undefined values are computed.

The metric is given by

$$SCD_{ki} = \sum_{j=1}^{n} \left(\sqrt{y_{kj}} - \sqrt{x_{ij}} \right)^2$$

All the functions were used individually to select the two most similar/nearest cases to impute values there by providing us with six different data sets. Distance metrics for each missing case were calculated using all the cases in the complete set and ranked in an ascending order. The first two minimum distances from the incomplete case being imputed, were selected as donors. For qualitative variables, one of two selected donors was picked randomly. For quantitative variables, the arithmetic mean of the values of the two donors was used to impute wherever necessary.

5.3.4.7. Combination Method

Finally, we devised a combination of two distance measures, which was used to compute the distance metric here. For each incomplete case, two metrics were calculated with respect to each complete case. One metric represented the categorical variables and the other represented the quantitative variables respectively. For a given incomplete case, Hamming distance was calculated which included only the dummy coded categorical variables with each of the cases from the complete case set. The Hamming distance between two sets of binary digits is the number of corresponding binary digit positions

that differ divided by the number of comparisons made. If two lists of binary values are compared, the Hamming distance is the number of items that are not identical. The metric is given by

$$HD_{ki} = \#(y_k \neq x_i)/\mathrm{n}$$

And, the Cosine Distance was computed which included only the quantitative variables. For each case in the complete set (with respect to an incomplete case), both the metrics were added and the summed up values were ranked in an ascending order. The cases with the first two smallest distances from the complete set were selected as donors. All the values were standardized using z score for SRPI and *k*-NN methods.

Software project data sets are made up of both categorical and quantitative variables. They also include interval scaled variables (categorical). "*Since adjustment cells are formed from the joint levels of categorical variables, they are not ideal for interval scaled variables*" [3]. Hence the Hot-Deck with adjustment cells was not used in our analysis.

5.3.5. Full Information Maximum Likelihood Approach

A method of estimating parameters of a given sample by the value that maximizes the likelihood of a sample. Maximum likelihood estimation begins with an expression known as a likelihood function for the sample data. The likelihood of a sample of data is the probability of obtaining that particular sample of data given the chosen probability model. This expression contains the unknown parameters. Those values of the parameter that maximize the sample likelihood are known as the maximum likelihood estimates [39, 40, 41, 42, 44]. Raw Maximum Likelihood Function (Full Information Maximum Likelihood) uses all the available data to generate a vector of means and a covariance

matrix among the variables. The FIML estimator maximizes the likelihood function, which is the sum of *m* case wise likelihood functions. A likelihood function is calculated for each individual that measures the change between the observed data for the j^{th} case and the current parameter estimates. The following function is maximized (assuming a multivariate normality distribution for the data set) [3, 7, 43]:

$$log\ L_j = K_j - \tfrac{1}{2} log|\Omega_j| - \tfrac{1}{2}(x_j - \mu_j)' \Omega^{-1}_j (x_j - \mu_j)$$

where x_j is the vector representing case *j*,

μ_j is the vector of mean estimates for variables observed for case *j*,

K_j is a constant that depends on the number of non-missing values for case *j*,

the determinant and inverse of Ω_j (Covariance Matrix) depend on variables that are observed for case *j*.

By adding the *m* case wise functions we get the log likelihood for the whole sample:

$$log\ L(\mu, \Omega) = \sum_{j=1}^{m} log\ L_j$$

It is a good technique to produce unbiased estimates when the data are MAR/MCAR. Studies have also shown that the method performs reasonably well when the conditions are non-ignorable too. It provides a consistent approach to parameter estimation problems and it has desirable mathematical properties. Popular statistical software packages provide use of maximum likelihood estimates. But ML approach is computationally demanding and it may be biased for small samples. We implemented the FIML method as described in [3, 7].

5.4 Experimental/Statistical Results

We used the following the measures of goodness of fit and accuracy.

5.4.1. Adjusted R-squared (Regression Correlation Coefficient)

It is the square of the correlation coefficient between the dependent variable and the estimate of it produced by the regressors. It is defined as the ratio of explained (regression) variation of the dependent variable to total variation. It has a value between 0 and 1 and if the value is close to 0, a poor model was built. When there are a large number of independent variables, R^2 may become large. It is therefore essential to adjust the value of R^2 as the number of independent variables increases.

5.4.2. Mean Magnitude of Relative Error

The impact of the imputation methods are then determined using Mean Magnitude of Relative Error. These statistics are calculated from the model built using the predicted data sets. Magnitude of Relative Error is defined as

$$MRE_i = (|Actual\ Effort_i - Estimated\ Effort_i|)/Actual\ Effort_i$$

Where "*i*" is the observed case.

This is estimated for all predicted observations and the mean of all these values gives us Mean Magnitude of Relative Error (MMRE).

5.4.3. Prediction at Level l (Pred(l))

$Pred(l) = p/n$ where p is the number of cases having relative error less than or equal to l and n is the total number of cases.

5.5 Performance Evaluation

Our primary aim was to investigate if prediction accuracies improved when completeness of a data set is enhanced using imputation techniques. We tried to

maximize the response in the data set for the same [1, 3]. We applied four missing data techniques to each of the six different data sets accumulated. The methods include Listwise Deletion (LD), Mean Imputation (MI), ten variants of Hot-Deck (HD) Imputation and Full Information Maximum Likelihood Approach (FIML).

The most common approach, LD was used in order to compare if other imputation methods performed better [1]. We used MI to test if simple techniques gave better prediction accuracies. We used HD variants because of their broad usage and proven performance [38, 45, 46, 47, 48, 49]. Finally, we used FIML [7, 43] in order to investigate their robustness under different conditions. The results show that we found a reasonable improvement in the prediction accuracies. The results of our experiment have shown that there was significant improvement in accuracy as well as fitting. The adjusted-R squared is a measure of goodness of fit and MMRE indicates accuracy. We now elaborate the impact of all the methods with respect to each data set taking into account their different inherent characteristics.

Table 3: Adjusted R – Squared, MMRE and Pred values of each of the methods for DS 1

Data Set 1	Adj R^2	MMRE	Pred(25%)
LD	.32	165%	21%
MI	.41	109%	19%
Sequential Hot-Deck	.43	74%	37%
Random Hot-Deck	.46	89%	23%
SRPI	.69	55%	46%
***Euclidean**	.72	61%	52%
***Manhattan**	.84	63%	41%
***Maholanobis**	.59	67%	39%
***Correlation**	.64	56%	47%
***Cosine**	.56	59%	54%
***Squared-Chord**	.71	50%	38%
***Combination Method**	.79	41%	59%
FIML	.8	42%	61%

5.5.1 Performance of the methods on Data Set 1 (DS1)

Based on our classification scheme, Data Set 1 (DS1) is a small sized data set having an ignorable missing mechanism (MAR), a missing data percentage <15% and has data missing arbitrarily. We can observe from table 3 that LD was inferior to all other methods. The reason would be the MAR mechanism. Moreover, only 7 cases were utilized by the method. Even though the total percentage of missing data was less than 15%, the total data loss was approximately 56% as the data set had only 7 complete cases. The Adj R^2 = .32 shows us the poor model built and the MMRE = 165% shows the bias in the estimates. The performance of LD deteriorates as the number of cases with missing values increase. This converse of the above statement does not necessarily be true as other factors to could influence its performance.

MI performed slightly better than LD but again the MAR condition accounted for its poor performance. Among the HD variants Sequential HD and Random HD performed inferior to the others (though they performed better than LD and MI). SRPI had a good Adj R^2 = .69 value and a better accuracy (MMRE = 55%). Within the *k*-NN HD variants, excluding Manhattan Distance Metric (Adj R^2 = .84 and MMRE = 63%) and Combination Method (Adj R^2 = .79 and MMRE = 41%), all of them performed more or less the same but with a better Adj R^2 and MMRE values than previous methods. Though the goodness of fit of the Manhattan Distance Metric is better than that of the Combination Method, the MMRE indicator shows that the Combination Method was much more accurate. The overall performance of the HD variants was better under MAR conditions. Finally, FIML (Adj R^2 = .8 and MMRE = 42%) performed well showing flexibility with small sized data sets.

5.5.2 Performance of the methods on Data Set 2 (DS2)

Table 4: Adjusted R – Squared, MMRE and Pred values of each of the methods for DS 2

Data Set 2	Adj R^2	MMRE	Pred(25%)
LD	.4	94%	18%
MI	.21	102%	9%
Sequential Hot-Deck	.11	114%	6%
Random Hot-Deck	.61	63%	33%
SRPI	.6	57%	34%
***Euclidean**	.69	61%	41%
***Manhattan**	.71	53%	44%
***Maholanobis**	.68	50%	49%
***Correlation**	.7	52%	47%
***Cosine**	.61	53%	40%
***Squared-Chord**	.66	67%	39%
***Combination Method**	.7	44%	48%
FIML	.72	46%	40%

Data Set 2 (DS2) is a small sized data set having an ignorable missing mechanism (MAR), a missing data percentage > 30% & < 45% and has data missing monotonously. We can observe from table 4 LD performed better than both MI and Sequential HD. The reason is due to the pattern in which the data are missing.

Both MI (MMRE = 102%) and Sequential HD (MMRE = 114%) showed high biases for the same reason. Because of the monotonous missing pattern, the same value was imputed in all the missing values for each variable using MI, thus distorting the distribution and underestimating variance. In the case of Sequential HD, the same donor was repeatedly used. Also the percentage of missing data could have played a role for the poor performance of MI. Random HD (Adj R^2 = .61 and MMRE = 63%) performed better in this case. SRPI too performed similar to Random HD. SRPI (Adj R^2 = .6 and MMRE = 57%) performed well in spite of the monotonous pattern. Among the *k*-NN HD variants, Manhattan Distance Metric (Adj R^2 = .71 and MMRE = 53%) and Combination

Method (Adj R^2 = .7 and MMRE = 44%) slightly outperformed others. FIML (Adj R^2 = .72 and MMRE = 46%) had the best fit for DS 2 among all the methods.

5.5.3 Performance of the methods on Data Set 3 (DS3)

Data Set 3 (DS3) is a small sized data set having an ignorable missing mechanism (MCAR), a missing data percentage <15% and has univariate missing data pattern. From table 5, we can see that LD (Adj R^2 = .79 and MMRE = 36%) performed very well under MCAR conditions. Under MCAR conditions, almost all the other methods too performed exceedingly well except for MI (Adj R^2 = .43 and MMRE = 71%) and Sequential HD (Adj R^2 = .5 and MMRE = 55%). The pattern of the missing values accounted for their underperformance. Under univariate missing pattern, the same donor values were used to impute among all the missing values, thus distorting the underlying distribution. Euclidean Distance Metric (Adj R^2 = .9 and MMRE = 30%), Correlation Distance Metric (Adj R^2 = .91 and MMRE = 28%) and the Combination Method (Adj R^2 = .9 and MMRE = 29%) performed better than the other methods giving the best fits and accuracies.

Table 5: Adjusted R – Squared, MMRE and Pred values of each of the methods for DS 3

Data Set 3	Adj R^2	MMRE	Pred(25%)
LD	.79	36%	58%
MI	.43	71%	15%
Sequential Hot-Deck	.5	55%	21%
Random Hot-Deck	.78	35%	52%
SRPI	.88	31%	61%
***Euclidean**	.9	30%	64%
***Manhattan**	.89	32%	65%
***Maholanobis**	.8	37%	60%
***Correlation**	.91	28%	71%
***Cosine**	.78	39%	65%
***Squared-Chord**	.88	32%	61%
***Combination Method**	.9	29%	74%
FIML	.87	32%	70%

5.5.4 Performance of the methods on Data Set 4 (DS4)

Table 6: Adjusted R – Squared, MMRE and Pred values of each of the methods for DS 4

Data Set 4	Adj R^2	MMRE	Pred(25%)
LD	.25	89%	16%
MI	.56	57%	24%
Sequential Hot-Deck	.51	64%	31%
Random Hot-Deck	.41	70%	19%
SRPI	.52	60%	40%
***Euclidean**	.61	50%	36%
***Manhattan**	.68	34%	35%
***Maholanobis**	.58	62%	37%
***Correlation**	.55	69%	42%
***Cosine**	.5	63%	41%
***Squared-Chord**	.49	73%	56%
***Combination Method**	.7	32%	68%
FIML	.6	36%	66%

Data Set 4 (DS4) is a medium sized data set having an ignorable missing mechanism (MAR), a missing data percentage >15% & < 30% and has data missing arbitrarily. From table 6, we can notice LD (Adj R^2 = .25 and MMRE = 89%) performed badly because only 9 cases were complete out of the total 42 cases in DS4. A total data loss of 79% was accounted while using LD. MI (Adj R^2 = .56 and MMRE = 57%), Sequential HD (Adj R^2 = .51 and MMRE = 64%) performed better than LD.

Though the missing data percentage was high MI and Sequential HD performed relatively well. SRPI and *k*-NN methods performed better than the LD, MI, Sequential HD or Random HD. Of them, Manhattan Distance Metric (Adj R^2 = .68 and MMRE = 34%) and Combination Method (Adj R^2 = .7 and MMRE = 32%) had the best fits and accuracies. Both of them performed better than FIML (Adj R^2 = .6 and MMRE = 36%). Overall, most of the HD variants performed similar to or better than FIML.

5.5.5 Performance of the methods on Data Set 5 (DS5)

Table 7: Adjusted R – Squared, MMRE and Pred values of each of the methods for DS 5

Data Set 5	Adj R^2	MMRE	Pred(25%)
LD	.1	1125%	4%
MI	.29	486%	9%
Sequential Hot-Deck	.16	986%	6%
Random Hot-Deck	.35	211%	12%
SRPI	.36	105%	16%
***Euclidean**	.4	89%	19%
***Manhattan**	.44	90%	21%
***Maholanobis**	.41	80%	20%
***Correlation**	.32	96%	18%
***Cosine**	.36	98%	23%
***Squared-Chord**	.38	103%	14%
***Combination Method**	.4	85%	22%
FIML	.52	55%	46%

Data Set 5 (DS5) is a medium sized data set having an ignorable missing mechanism (MAR), a missing data percentage > 45% and has data missing arbitrarily. Looking at table 7, we can see that all the methods performed badly. None of them gave a reasonable accuracy. None of them had a reasonable goodness of fit. LD (Adj R^2 = .1 and MMRE = 1125%) performed the worst of all. The HD variants performed more or less the same. The reason for such a performance by all the methods is because of the high percentage of missing data (46%). With a huge amount of data missing, none of the methods could lessen bias.

5.5.6 Performance of the methods on Data Set 6 (DS6)

Data Set 6 (DS6) is a large sized data set having a non-ignorable missing mechanism (NI), a missing data percentage >15% & < 30% and has data missing arbitrarily. We can notice from table 8 that neither LD (Adj R^2 = .21 and MMRE = 218%) nor MI (Adj R^2 = .35 and MMRE = 109%) performed well under NI conditions.

Sequential HD (Adj R^2 = .4 and MMRE = 87%) and Random HD (Adj R^2 = .41 and MMRE = 84%) were slightly better than the previous two but both of them underperformed as well. SRPI and all *k*-NN methods had Adj R^2 values around .5 to .6 and MMRE values between 55%-70%. Manhattan Distance Metric (Adj R^2 = .58 and MMRE = 70%), Squared-Chord Distance Metric (Adj R^2 = .58 and MMRE = 65%) and Combination Distance Metric (Adj R^2 = .59 and MMRE = 57%) had better accuracies among them.

Table 8: Adjusted R – Squared, MMRE and Pred values of each of the methods for DS 6

Data Set 6	Adj R^2	MMRE	Pred(25%)
LD	.21	218%	6%
MI	.35	109%	11%
Sequential Hot-Deck	.4	87%	15%
Random Hot-Deck	.41	84%	14%
SRPI	.5	68%	13%
***Euclidean**	.52	63%	21%
***Manhattan**	.58	70%	23%
***Maholanobis**	.54	66%	24%
***Correlation**	.52	60%	29%
***Cosine**	.5	64%	31%
***Squared-Chord**	.58	65%	24%
***Combination Method**	.59	57%	30%
FIML	.67	48%	56%

It was FIML (Adj R^2 = .67 and MMRE = 48%) that was most resilient to bias under non-ignorable missing mechanism conditions. FIML had the least bias and best estimates of all the methods under NI conditions. It was the only method that built a reasonable model and accuracy under NI conditions. Fig 8 depicts the performance of each method on the six data sets. Mean Magnitude of Relate Error (MMRE) indicates the accuracy with which the estimates were predicted from the effort prediction models. Each graph represents each imputation method implemented on each of the six data sets.

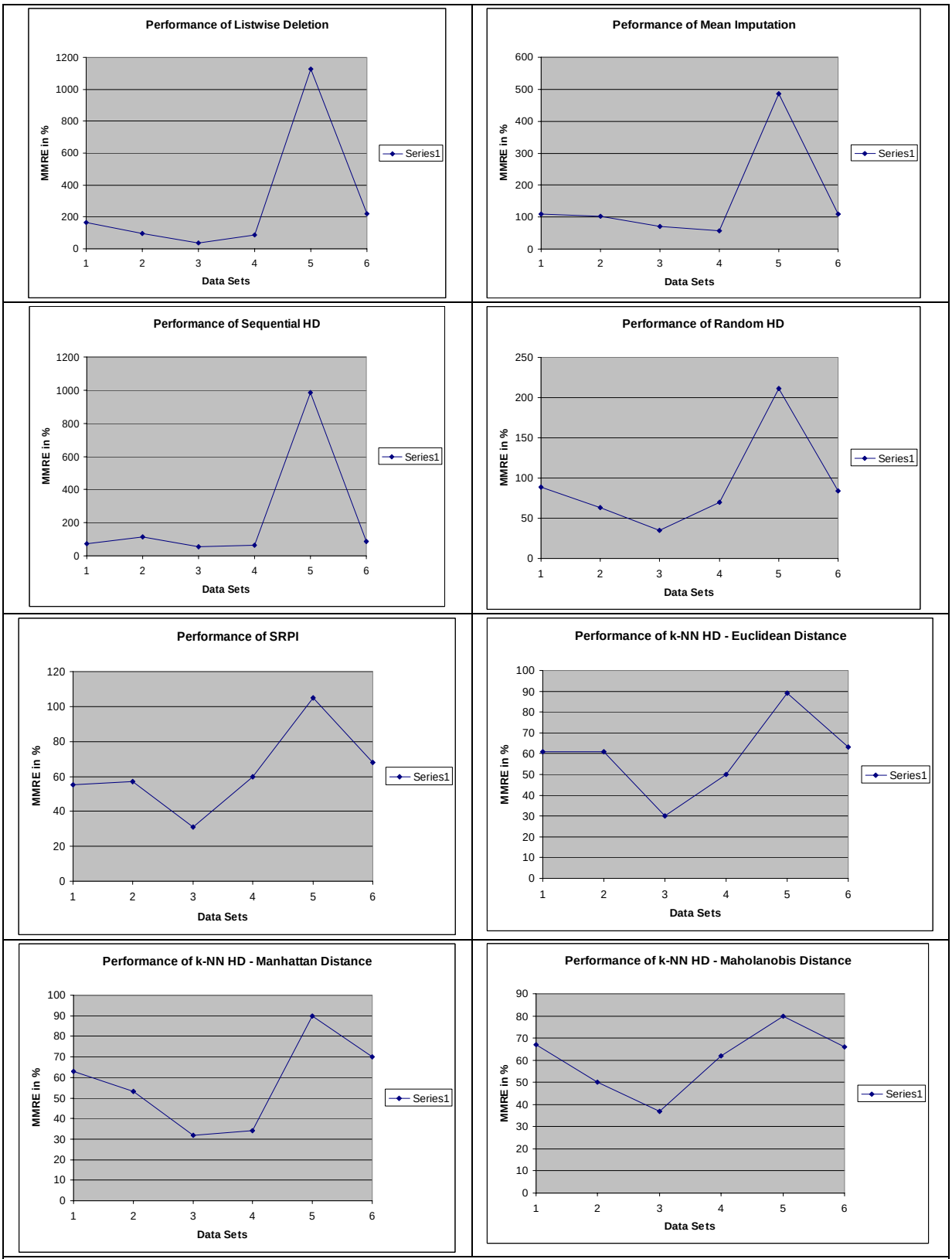
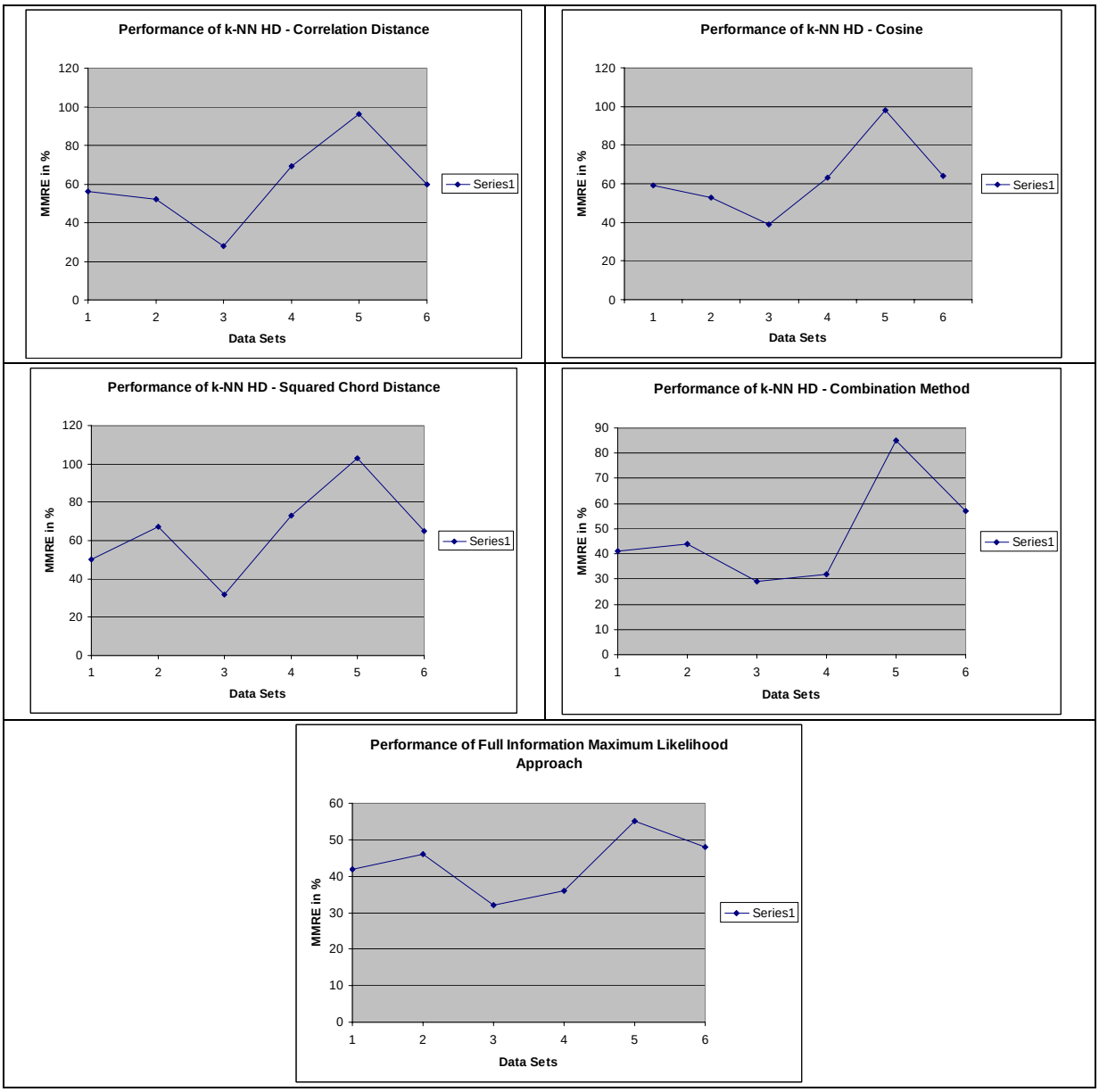


Fig 8: Performances (Accuracy Measures - Mean Magnitude of Relate Error (MMRE)) of each of the Imputation Methods w.r.t the 6 data sets

(fig 8 cont.)


Performance of k-NN HD - Correlation Distance
MMRE in %
Data Sets
Series1
Performance of k-NN HD - Cosine
MMRE in %
Data Sets
Series1
Performance of k-NN HD - Squared Chord Distance
MMRE in %
Data Sets
Series1
Performance of k-NN HD - Combination Method
MMRE in %
Data Sets
Series1
Performance of Full Information Maximum Likelihood Approach
MMRE in %
Data Sets
Series1

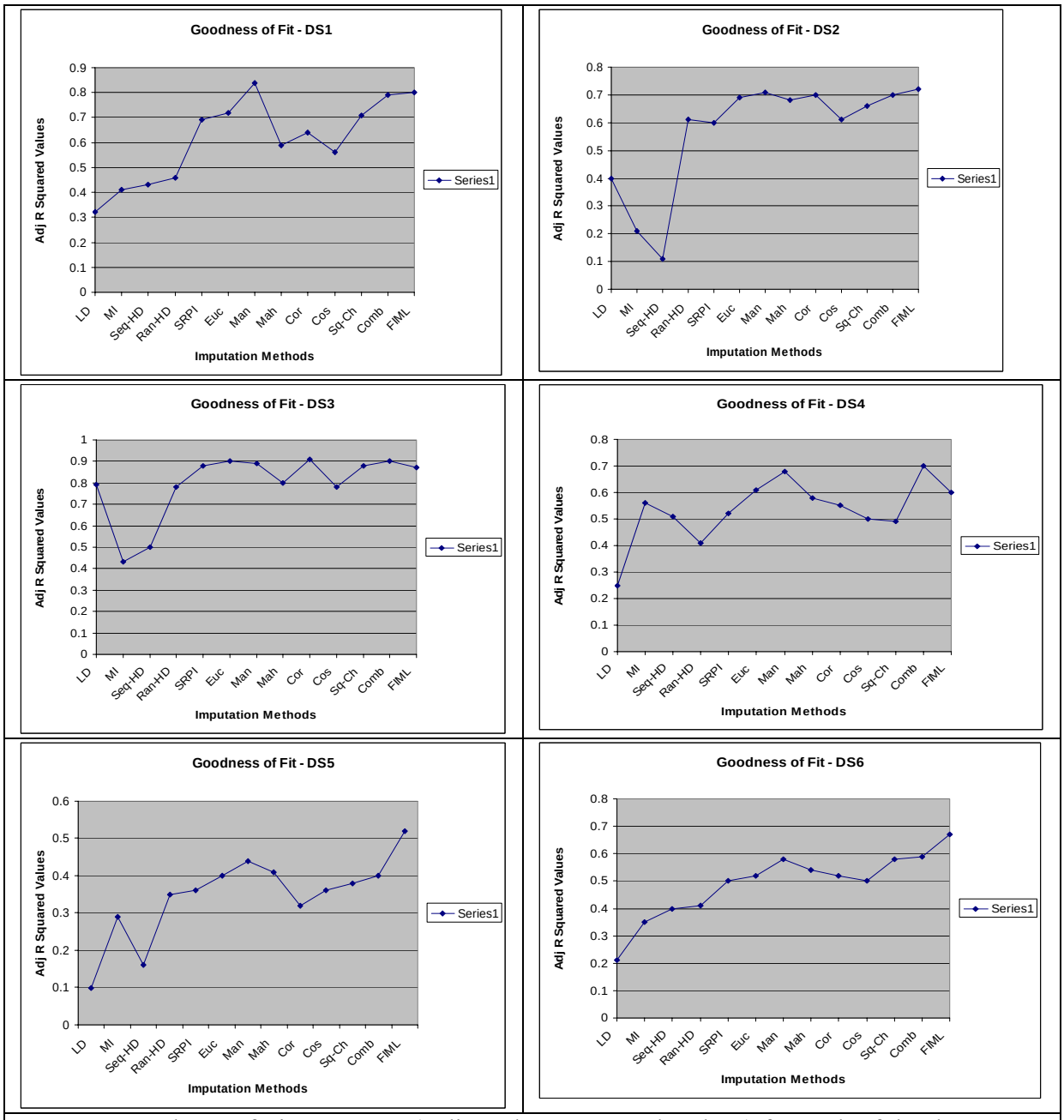


Fig 9: Goodness of Fit Measures (Adjusted – R Squared Values) for each of the data sets using the Imputation Methods

Fig 9 represents the quality of the models built by each method when implemented on each of the six data sets. The adjusted-R squared is a measure of goodness of fit. It has a value between 0 and 1 and if the value is close to 0, it means a poor model was built. The Adjusted R-Squared values of each of the methods on each of the data sets explain the impact of that particular method. It tells us the quality of the effort prediction model built by the data set imputed from that particular method.

Each graph represents each data set and the methods are represented on the x-axis. The Manhattan Distance Method, the Maholanobis Distance Method, the Combination Method and FIML performed consistently. There were a few instances when other distance methods (Correlation, Cosine) built good models though.

Fig 10 represents the Pred (25%) values by each method when implemented on each of the six data sets. It represents the percentage of cases having relative error less than or equal to *l* (*l* is 25% in our study) and *n* is the total number of cases. It is an index measure that explains to us what percentage of cases with in a data set had predicted estimates with a relative error less than 25%. High prediction values indicate that the imputation method performed well in generating unbiased estimates.

In some analyses, such information could be very significant particularly to point out outliers or extreme cases. Each graph represents each data set and the methods are represented on the x-axis. The y-axis denotes the pred values in %. The best pred values were recorded for the Combination Method and the FIML. Both generated high percentage estimates that had low relative error. More than 75% of cases had biases more than the relative error for Listwise Deletion. Most of the methods did not achieve high pred values when compared with the Combination Method and FIML.

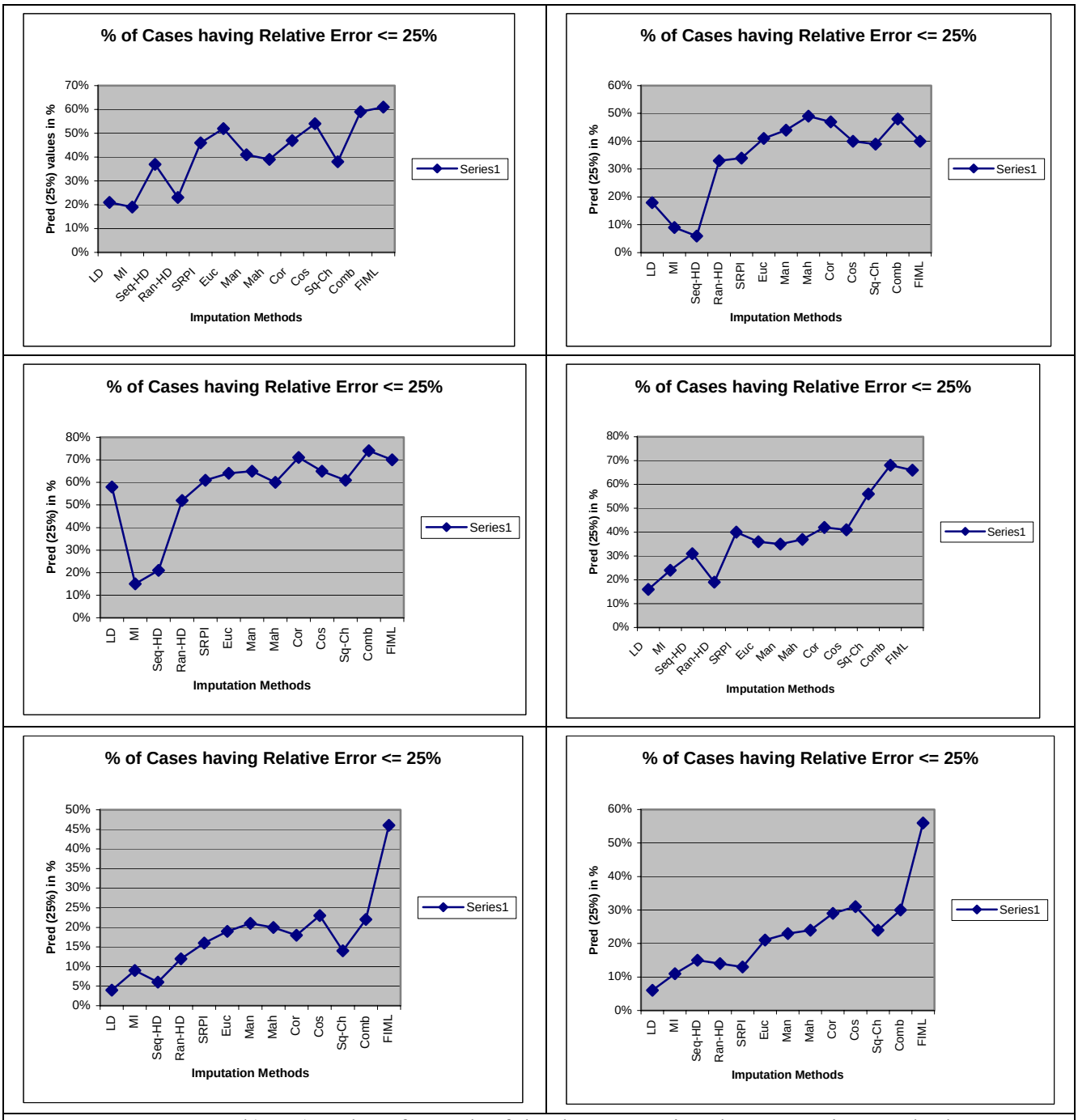


Fig 10: Pred(25%) Values for each of the data sets using the Imputation Methods

CHAPTER 6 PROPOSED METHODOLOGY

In this chapter, we describe our hybrid methodology, which proceeds data imputation in two phases. The data set is first divided into two sets, complete set and incomplete set. The complete set has cases with no missing values and the incomplete set has all the cases with missing values. In the first phase, the complete data set is processed to form homogeneous clusters. A hierarchical agglomerative clustering algorithmic approach is used to form clusters which contain one or more "similar" cases. Once the clusters are formed, missing values for each case in the incomplete set are imputed by selecting donors from that particular cluster that they most probably would belong to. The cluster that would contribute the donor is determined by calculating a proximity metric for each missing case.

Initially, the data set is divided into two sets, one having cases with no missing values and the other having only cases with missing values. The complete set having cases with no missing values are then classified into clusters using a hierarchical agglomerative clustering algorithm. Hierarchical agglomerative clustering algorithms are bottom-up approaches in which each case is considered an individual cluster and at each step, the most similar pair of clusters is merged together. Cluster similarity is calculated using a distance measure.

6.1 First Phase

6.1.1. A Simple Agglomerative Clustering Algorithm

An agglomerative clustering algorithm is a bottom-up approach, which is used to form homogenous clusters. The following are the steps in the agglomerative hierarchical clustering algorithm for grouping *n* cases:

1. Start with *n* clusters (each cluster representing each of the *n* cases in the data set). An *n* x *n* symmetric similarity matrix between all pairs of clusters is generated. The matrix represents similarity between the clusters. The ij^{th} entry in the matrix gives the similarity between the i^{th} and j^{th} clusters. The similarity is measured using a distance metric.

2. The similarity matrix is searched for the most similar (the pair having the smallest distance value) pair of clusters in order to merge them. Let us suppose the distance between "most similar" clusters *i* and *j* be *d*. Merge *i* and *j*. Note the newly generated cluster as (*ij)*. Update the entries in the distance matrix to reflect the pairwise similarity between the newly formed cluster and the original clusters by:

   a. first deleting the rows and columns representing clusters *i* and *j* and

   b. next adding a row and column denoting the distances between newly formed cluster (*ij)* and the remaining clusters.

3. Repeat the steps a and b until a single cluster containing all the cases would be formed. The algorithm terminates after (*n-1*) times. The identities of the clusters that are merged are recorded. Also the levels at which the mergers take place are noted. The complexity of the algorithm is $O(n^2 logn)$.

Different variants of agglomerative hierarchical clustering algorithms are formed based on the way the similarity is measured between the clusters. Average Linkage Agglomerative Clustering Algorithm was used in our approach. In this method, the distance between two clusters is defined as the average of distances between all pairs of cases, where each pair is made up of one case from each group. The average distance *d(i,j)* computed at each level is given by the following equation:

$d(i, j)$ is computed as $d(i, j) = D_{ij} / (n_r * n_s)$

Where D_{ij} is the sum of all pairwise distances between cluster i and cluster j. n_r and n_s are the sizes of the clusters i and j respectively. At each stage of hierarchical clustering, the clusters i and j, for which d_{ij} is the minimum, are merged. The distance D measured is the Euclidean Distance. The figure below illustrates average linkage clustering:

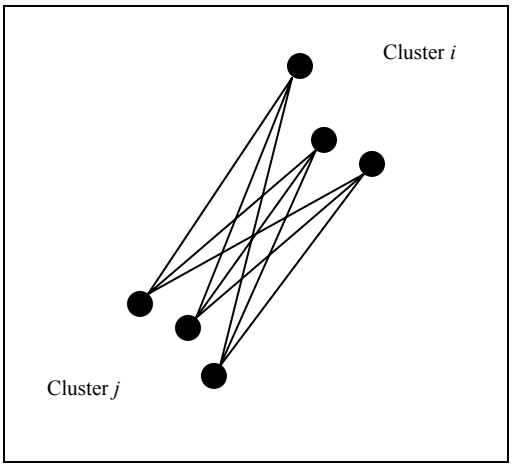


Fig 11: Average of pairwise distances calculated between cases of two clusters

6.1.2. Concept of "United Clusters"

We use the concept of "united clusters" specified as "mutual clusters" in [60] which contain a group of cases that are sufficiently close to each other and far from all other cases and hence should never be separated. The united clusters are mutual clusters and in no way are different. We termed them united clusters, as it seemed more appropriate.

The characteristic of a united cluster is atomicity. The definition of a united cluster is formally defined by [60], as a group G of cases such that

$$\forall\ c_i \in G\ \&\ c_j \notin G, D(c_i, c_j)\ > \text{Diameter}(G)\ \{= \text{Maximum}\ D(c_p, c_q)\ (\text{where}\ c_p, c_q \in G)\}$$

Where c_i, c_j, c_p, c_q are cases and D is the distance.

It means that the largest distance between the cases in *G* be less than the smallest distance from a case in *G* to any case not in *G*. Such cases are grouped as a united cluster. And they remain in the same cluster and are not to be separated. For the purpose, such clusters are identified prior to running the clustering algorithm. Each of such groups are represented by a single case which is basically the centroid vector formed from all the members of such a group. Given a group of cases represented by *G,* their corresponding vector representations, we define the centroid vector ***c*** to be

$$c = 1/(|\text{No. of cases in } G|) \sum d \text{ (where } d \text{ belongs to } G)$$

which is nothing more than the vector obtained by averaging the values of the various variables present in the cases of *G.* Once the clustering algorithm is run and we decide on the number of clusters to be formed, we reassign the clusters containing each of the centroid vectors with the original members of the united cluster represented by the centroids. The idea is explained in more detail by Chipman and Tibshirani in [60]. We implemented the same in our methodology.

As hierarchical agglomerative algorithms start with all cases, they are particularly effective in identifying many small clusters. The quality of clusters deteriorates as more number of mergers is made and hence with such a concept of united clusters, we essentially decrease the number of mergers, thereby increasing cluster quality.

The algorithm first observes that any two nearest cases in a united cluster must be a united pair. As no other case is closer to these two cases, they form a united pair. The algorithm starts by identifying such pairs and then would use these pairs as initial seeds to find all united clusters. UP denotes the set of all united pairs, and UC denotes the set of all united clusters. The set W is the temporary cluster used in the algorithm to find larger

clusters. The algorithm finally finds larger clusters by identifying a case c that will least increase the diameter (i.e. maximum within-group distance) of the resultant set W $\cup$ c. If that new set is a united cluster, it is appended to the set UC. We implemented the following algorithm, which is elaborated in [60] to identify the united clusters.

Step 1: The set of all united pairs is found. UP is a null set initially.

For i = 1 to n

For each of the n cases in the data set, find j such that it is the nearest case to the case i

If $d_{ij} < d_{kj}$ and $k \neq i, j$ then UP $\leftarrow$ UP $\cup$ $\{i, j\}$

It finds that any two nearest cases in a united cluster must be a united pair. If d_{ij} is the distance between i and j and if it is less than the distance between j and any other case in the data set (other than i and itself), then the algorithm identifies $\{i, j\}$ as a united pair.

Step 2: All the united pairs in the set UP, are then assigned to the set of United Clusters, UC. Each of the united pairs is used as initial seeds in order to identify the united clusters.

UC $\leftarrow$ UP

Step 3: For i = 1 to |UP| (Number of united pairs formed)

Assign each united pair to the temporary cluster W

W $\leftarrow$ UP_i

For p = | UP_i | + 1 to n

Find a case c outside W such that when c is included in W, it would result the smallest increase in the diameter (maximum distance) of W.

$W \leftarrow W \cup c$

If the maximum distance d_{ij} (where i, j belong to W) < the minimum distance of d_{ik} (where i belongs to W and k does not belong to W), then UC $\leftarrow$ UC $\cup$ {W}. (The newly formed cluster W is checked to see if it still holds the property of a United Cluster.)

Step 4: UC represents all the united clusters in the data set.

6.1.3. Measuring the quality of the clusters formed

The measures of quality let us evaluate how well the clustering has been performed. We use three measures to quantify the quality of the clusters. The measure lets us evaluate how well the clustering is done. The measures are oriented towards measuring the effectiveness of the approach [61]. A performance measure of the clusters is to examine whether any of the cases were assigned wrongly to a cluster. To verify this, centroid vectors are calculated for each of the cluster initially. Next, cases having smaller distance to centroid vectors of other clusters when compared to the centroid vector of the cluster they belong to are gathered. Such cases are considered to be wrongly assigned to a cluster.

The next measure is within-cluster distances, which provides a measure of "goodness" for the clusters. It identifies clusters that have minimum within-cluster distances. After the clustering algorithm is run, the dendrogram representing the sequence of partitions of the data set is cut at different levels to form varying numbers of clusters. For each number of clusters, the sum of all pairwise within-cluster (Euclidean Distance) distances is calculated. Good clusterings minimize the sum. The sum of distances is plotted versus the number of clusters formed. Our approach does well when there is more

number of clusters. One final measure is to look at the inter-cluster dissimilarity in order to find how different are each of the clusters. For each of the clusters, a centroid vector is initially calculated and the pairwise cosine similarity is measured between all of them. The cosine formula is cos (d_i, d_j) = d_i^t *d_j where d_i and d_j are centroids vectors of two clusters of unit length. The measure is 1 if the centroids are identical and 0 if they are orthogonal. Inter-cluster dissimilarity should be high for good clustering quality.

6.2 Second Phase

After the clusters have been formulated, the imputation of missing values in each case within the incomplete set is performed. For each case, the proximity metric is calculated with respect to each cluster. This is done by first calculating the centroid vector for each cluster. The Euclidean distance between each missing case and the centroid of each cluster gives the proximity metric. The cluster representing the centroid vector with which the incomplete case gives the smallest proximity metric value is selected. From within the cluster, the donors are selected using the *k*-Nearest Neighborhood (Combination Method). The method works by finding "*k*" most similar/nearest complete cases to the incomplete case where the similarity is measured by a distance parameter (Cosine Distance (Quantitative Variables) and Hamming Distance (Qualitative Variables)). The value of "*k*" was set to 2. That is, 2 most similar/nearest cases were selected to impute the values in the incomplete case.

6.3 Experimental Results

Our goal was to check if the methodology performed better than the different imputation methods implemented in chapter 5. We implemented the methodology on the same six real-time datasets to evaluate its performance. The results showed significant

improvement in terms of prediction accuracies. We used the same metrics (adjusted-R squared, MMRE, Pred (25%)) to measure the goodness of fit of the model built from the imputed data set using our methodology and percentage of error/bias in the estimates.

The performance of the methodology can be quantified by taking a look at the statistics recorded. Prior to this, we would like to discuss the quality of the clusters formed using the clustering algorithm in the first phase of the methodology. The cluster quality was measured using three metrics. Both within-cluster-similarity and inter-cluster dissimilarity were measured to get a better look at the quality of the clustering achieved.

6.3.1. Measures of Cluster Quality

The measures of quality let us evaluate the performance of the clustering technique. We used three measures.

6.3.1.1. Miss-assignment Count

The first measure finds the number of cases that have been incorrectly assigned to a cluster. The distances of all the cases from the centroids of all the clusters are calculated. Also the distances of all the cases from the centroid of the cluster to which they are member of are calculated. Any case having a distance greater from the centroid of the cluster to which it belongs when compared to its distance from a centroid of another cluster is designated a miss-assigned case.

Table 9: Number of missassigned cases in each data set for varying number of clusters

	5	**10**	**12**	**25**	**40**	**55**	**80**	**93**
DS 1	0	1	0	-	-	-	-	-
DS 2	0	1	0	0	-	-	-	-
DS 3	0	0	0	-	-	-	-	-
DS 4	1	2	0	0	0	-	-	-
DS 5	4	2	1	0	0	0	-	-
DS 6	7	5	4	2	2	1	0	0

Table 9 shows wrongly assigned cases for each of the six data sets. The rows indicate each of the six data sets and the columns indicate the number of clusters taken into account. The number of miss-assigned points is only one for DS 1 and DS 2. The method performed well with no miss-assigned points for DS 3. DS 4 had one miss-assigned point when five clusters were formed and had two miss-assigned points when ten clusters were formed. DS 5 had four miss-assigned points when five clusters were formed, two miss-assigned points when ten clusters were formed, and one miss-assigned point when twelve clusters were formed. The number of miss-assignments is low for the first 5 data sets when compared to DS 6. The number of miss-assigned cases is highest for DS 6 where missing mechanism is under NI conditions. DS 6 had seven miss-assigned points when five clusters were formed, five miss-assigned points when ten clusters were formed, four miss-assigned points when twelve clusters were formed, two miss-assigned points when twenty-five clusters were formed, two miss-assigned points when forty clusters were formed and one miss-assigned point when fifty-five clusters were formed. The clustering algorithm used is a bottom-up approach, which starts the process of clustering by considering each case a new "cluster", and thus forth proceeds. The very nature of this approach makes it a good method in identifying a large number of small clusters, which is highly significant. The overall number of miss-assigned points can be considered low. Even under NI conditions (for DS 6) the number of miss-assigned points was low when twenty-five or more clusters were formed.

6.3.1.2. Within-Cluster Distances

It provides a measure of "goodness" for the clusters by identifying the minimum within-cluster (square root of the sum of squared) distances. For varying number of

clusters, the sum of all pairwise within-cluster distances is calculated. Minimized sums represent good clusterings, which means that the cases within a cluster are closer to each other. This metric measures the similarity between the cases in a cluster, said otherwise within-cluster similarity. The proposed approach does well when there is reasonably higher number of clusters. The graphs plotted below shows us the same.

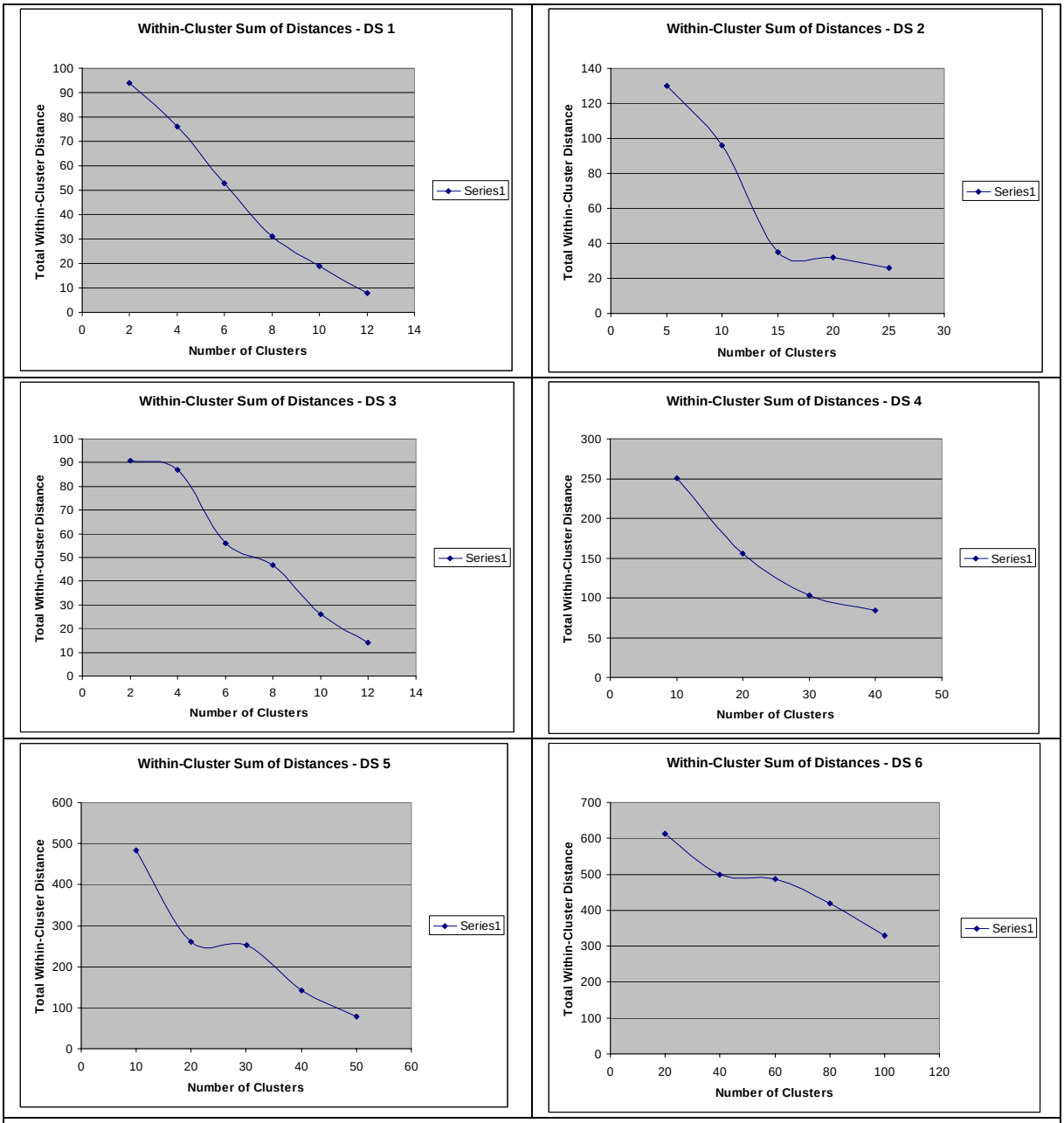


Fig 12: Within-Cluster Sum of Distances for each data set for varying number of clusters

6.3.1.3. Inter-Cluster Dissimilarity

It measures the dissimilarity between the clusters formed. The more dissimilar the clusters are, the better the quality of clustering accomplished. The dissimilarity between two clusters is calculated by finding the cosine similarity between their centroids. As the similarity between the clusters increases, the value of the metric approaches 1 and as the dissimilarity increases, the value approaches 0. The degree of orthogonality between the centroids is what the metric calculates. The decision on how many clusters would be ideal was made using the first metric, miss-assignment count. For data sets 1, 2, 3, and 4, 12 clusters were formed and for datasets 5 and 6, 25 clusters were formed. The first table shows the similarity values for all the cluster pairs formed in the data sets 1, 2, 3, and 4. The second table depicts the values of data set 5 and the third table that of data set 6. The values ranged from .01-.7, and only 28 readings recorded had values more than .5. This suggests that the clusters were different from one another and hence satisfactory clustering was achieved.

Table 10: Inter-Cluster Dissimilarities measured using Cosine Metric for Data Sets 1 to 4

Cluster Pair	DS 1	DS 2	DS 3	DS 4
1-2	.11	.09	.02	.1
1-3	.2	.12	.09	.09
1-4	.32	.13	.13	.15
1-5	.06	.26	.06	.38
1-6	.12	.05	.08	.25
1-7	.24	.03	.11	.06
1-8	.31	.09	.22	.21
1-9	.27	.28	.25	.16
1-10	.14	.39	.05	.09
1-11	.16	.42	.06	.28
1-12	.13	.12	.18	.17
2-3	.26	.18	.06	.05
2-4	.4	.11	.19	.41
2-5	.32	.2	.17	.52
2-6	.16	.09	.16	.19
2-7	.29	.05	.06	.26
2-8	.17	.07	.01	.09

(table 10 cont.)

2-9	.09	.02	.09	.13
2-10	.1	.25	.04	.05
2-11	.23	.36	.25	.12
2-12	.33	.01	.24	.27
3-4	.6	.17	.19	.35
3-5	.19	.28	.13	.36
3-6	.15	.02	.18	.05
3-7	.35	.09	.04	.4
3-8	.25	.12	.06	.25
3-9	.04	.06	.09	.19
3-10	.09	.28	.18	.07
3-11	.11	.41	.11	.02
3-12	.21	.08	.2	.19
4-5	.03	.06	.01	.16
4-6	.16	.08	.06	.23
4-7	.42	.15	.16	.35
4-8	.12	.17	.12	.47
4-9	.06	.29	.2	.21
4-10	.14	.02	.1	.25
4-11	.08	.01	.08	.38
4-12	.36	.13	.12	.08
5-6	.3	.2	.15	.07
5-7	.02	.11	.07	.17
5-8	.22	.08	.06	.19
5-9	.01	.06	.13	.08
5-10	.26	.05	.24	.05
5-11	.09	.04	.01	.17
5-12	.05	.25	.14	.34
6-7	.23	.26	.21	.18
6-8	.06	.34	.17	.09
6-9	.14	.5	.03	.2
6-10	.01	.21	.08	.36
6-11	.34	.11	.15	.08
6-12	.21	.19	.04	.19
7-8	.04	.06	.06	.32
7-9	.11	.03	.11	.15
7-10	.1	.35	.25	.09
7-11	.29	.37	.18	.31
7-12	.05	.21	.14	.11
8-9	.01	.07	.09	.4
8-10	.22	.19	.21	.07
8-11	.35	.23	.32	.02
8-12	.06	.25	.19	.11
9-10	.28	.16	.15	.29
9-11	.08	.05	.13	.33
9-12	.07	.09	.22	.2
10-11	.02	.1	.1	.15
10-12	.14	.46	.09	.21
11-12	.07	.22	.06	.32

Table 11: Inter-Cluster Dissimilarities measured using Cosine Metric for Data Set 5

DS 5							
Cluster Pair	**Value**	**Cluster Pair**	**Value**	**Cluster Pair**	**Value**	**Cluster Pair**	**Value**
1-2	.23	5-6	.7	10-11	.25	17-24	.42
1-3	.31	5-7	.21	10-12	.19	17-25	.12
1-4	.16	5-8	.04	10-13	.07	18-19	.06
1-5	.09	5-9	.21	10-14	.02	18-20	.14
1-6	.42	5-10	.46	10-15	.5	18-21	.08
1-7	.5	5-11	.31	10-16	.3	18-22	.36
1-8	.41	5-12	.28	10-17	.29	18-23	.13
1-9	.3	5-13	.07	10-18	.18	18-24	.18
1-10	.25	5-14	.15	10-19	.3	18-25	.04
1-11	.11	5-15	.25	10-20	.08	19-20	.06
1-12	.09	5-16	.31	10-21	.07	19-21	.09
1-13	.5	5-17	.39	10-22	.25	19-22	.18
1-14	.3	5-18	.05	10-23	.31	19-23	.11
1-15	.29	5-19	.2	10-24	.39	19-24	.2
1-16	.18	5-20	.07	10-25	.05	19-25	.01
1-17	.3	5-21	.06	11-12	.2	20-21	.06
1-18	.08	5-22	.21	11-13	.07	20-22	.35
1-19	.07	5-23	.26	11-14	.14	20-23	.47
1-20	.17	5-24	.29	11-15	.01	20-24	.21
1-21	.25	5-25	.15	11-16	.34	20-25	.25
1-22	.1	6-7	.37	11-17	.21	21-22	.38
1-23	.24	6-8	.4	11-18	.04	21-23	.08
1-24	.46	6-9	.58	11-19	.11	21-24	.07
1-25	.45	6-10	.3	11-20	.21	21-25	.17
2-3	.28	6-11	.24	11-21	.32	22-23	.19
2-4	.19	6-12	.04	11-22	.19	22-24	.08
2-5	.2	6-13	.18	11-23	.15	22-25	.05
2-6	.08	6-14	.06	11-24	.13	23-24	.45
2-7	.16	6-15	.07	11-25	.22	23-25	.37
2-8	.13	6-16	.19	12-13	.24	24-25	.38
2-9	.33	6-17	.12	12-14	.46		
2-10	.21	6-18	.43	12-15	.45		
2-11	.19	6-19	.35	12-16	.28		
2-12	.49	6-20	.12	12-17	.19		
2-13	.35	6-21	.26	12-18	.25		
2-14	.27	6-22	.5	12-19	.05		
2-15	.34	6-23	.32	12-20	.06		
2-16	.14	6-24	.41	12-21	.18		
2-17	.16	6-25	.08	12-22	.06		
2-18	.59	7-8	.02	12-23	.16		
2-19	.45	7-9	.19	12-24	.59		
2-20	.32	7-10	.23	12-25	.45		
2-21	.22	7-11	.36	13-14	.32		
2-22	.19	7-12	.17	13-15	.22		

(table 11 cont.)

2-23	.08	7-13	.02	13-16	.19		
2-24	.21	7-14	.06	13-17	.08		
2-25	.08	7-15	.28	13-18	.05		
3-4	.01	7-16	.4	13-19	.12		
3-5	.06	7-17	.26	13-20	.27		
3-6	.29	7-18	.19	13-21	.35		
3-7	.33	7-19	.13	13-22	.36		
3-8	.37	7-20	.08	13-23	.05		
3-9	.14	7-21	.16	13-24	.4		
3-10	.23	7-22	.25	13-25	.25		
3-11	.19	7-23	.37	14-15	.19		
3-12	.04	7-24	.4	14-16	.07		
3-13	.03	7-25	.56	14-17	.02		
3-14	.4	8-9	.24	14-18	.12		
3-15	.09	8-10	.19	14-19	.06		
3-16	.05	8-11	.17	14-20	.28		
3-17	.15	8-12	.08	14-21	.41		
3-18	.24	8-13	.05	14-22	.08		
3-19	.2	8-14	.26	14-23	.06		
3-20	.4	8-15	.19	14-24	.08		
3-21	.36	8-16	.34	14-25	.15		
3-22	.27	8-17	.21	15-16	.17		
3-23	.12	8-18	.17	15-17	.29		
3-24	.01	8-19	.37	15-18	.02		
3-25	.09	8-20	.45	15-19	.01		
4-5	.56	8-21	.29	15-20	.13		
4-6	.21	8-22	.08	15-21	.42		
4-7	.63	8-23	.29	15-22	.36		
4-8	.18	8-24	.17	15-23	.19		
4-9	.27	8-25	.02	15-24	.5		
4-10	.05	9-10	.05	15-25	.36		
4-11	.07	9-11	.14	16-17	.09		
4-12	.13	9-12	.29	16-18	.59		
4-13	.27	9-13	.33	16-19	.38		
4-14	.35	9-14	.44	16-20	.26		
4-15	.14	9-15	.5	16-21	.7		
4-16	.64	9-16	.02	16-22	.29		
4-17	.23	9-17	.45	16-23	.09		
4-18	.15	9-18	.29	16-24	.08		
4-19	.36	9-19	.34	15-25	.07		
4-20	.24	9-20	.21	17-18	.17		
4-21	.29	9-21	.08	17-19	.25		
4-22	.16	9-22	.04	17-20	.1		
4-23	.22	9-23	.16	17-21	.24		
4-24	.11	9-24	.29	17-22	.46		
4-25	.03	9-25	.38	17-23	.45		

Table 12: Inter-Cluster Dissimilarities measured using Cosine Metric for Data Set 6

DS 6							
Cluster Pair	**Value**	**Cluster Pair**	**Value**	**Cluster Pair**	**Value**	**Cluster Pair**	**Value**
1-2	.7	5-6	.34	10-11	.17	17-24	.23
1-3	.23	5-7	.21	10-12	.02	17-25	.35
1-4	.6	5-8	.04	10-13	.05	18-19	.47
1-5	.02	5-9	.11	10-14	.14	18-20	.21
1-6	.08	5-10	.21	10-15	.29	18-21	.25
1-7	.14	5-11	.32	10-16	.33	18-22	.38
1-8	.16	5-12	.19	10-17	.44	18-23	.08
1-9	.59	5-13	.15	10-18	.5	18-24	.07
1-10	.45	5-14	.13	10-19	.02	18-25	.17
1-11	.32	5-15	.24	10-20	.45	19-20	.19
1-12	.22	5-16	.19	10-21	.25	19-21	.08
1-13	.19	5-17	.13	10-22	.06	19-22	.05
1-14	.08	5-18	.18	10-23	.18	19-23	.17
1-15	.21	5-19	.04	10-24	.12	19-24	.34
1-16	.08	5-20	.06	10-25	.49	19-25	.18
1-17	.01	5-21	.09	11-12	.37	20-21	.09
1-18	.06	5-22	.18	11-13	.21	20-22	.2
1-19	.29	5-23	.11	11-14	.11	20-23	.36
1-20	.33	5-24	.2	11-15	.26	20-24	.08
1-21	.37	5-25	.01	11-16	.19	20-25	.19
1-22	.14	6-7	.06	11-17	.13	21-22	.1
1-23	.23	6-8	.16	11-18	.08	21-23	.09
1-24	.19	6-9	.12	11-19	.16	21-24	.15
1-25	.04	6-10	.2	11-20	.25	21-25	.38
2-3	.03	6-11	.1	11-21	.37	22-23	.25
2-4	.4	6-12	.08	11-22	.4	22-24	.06
2-5	.09	6-13	.12	11-23	.56	22-25	.21
2-6	.05	6-14	.15	11-24	.24	23-24	.16
2-7	.15	6-15	.07	11-25	.19	23-25	.09
2-8	.24	6-16	.06	12-13	.17	24-25	.28
2-9	.2	6-17	.13	12-14	.08		
2-10	.4	6-18	.24	12-15	.05		
2-11	.36	6-19	.01	12-16	.26		
2-12	.21	6-20	.03	12-17	.19		
2-13	.04	6-21	.09	12-18	.34		
2-14	.21	6-22	.28	12-19	.17		
2-15	.46	6-23	.39	12-20	.05		
2-16	.31	6-24	.42	12-21	.41		
2-17	.28	6-25	.12	12-22	.52		
2-18	.07	7-8	.18	12-23	.19		
2-19	.15	7-9	.11	12-24	.26		
2-20	.25	7-10	.2	12-25	.32		
2-21	.31	7-11	.09	13-14	.06		
2-22	.39	7-12	.05	13-15	.12		
2-23	.05	7-13	.07	13-16	.39		
2-24	.2	7-14	.02	13-17	.4		
2-25	.07	7-15	.19	13-18	.15		
3-4	.06	7-16	.27	13-19	.08		

(table 12 cont.)

3-5	.21	7-17	.06	13-20	.19		
3-6	.26	7-18	.09	13-21	.17		
3-7	.29	7-19	.18	13-22	.03		
3-8	.15	7-20	.11	13-23	.4		
3-9	.37	7-21	.2	13-24	.32		
3-10	.4	7-22	.01	13-25	.16		
3-11	.58	7-23	.06	14-15	.29		
3-12	.11	7-24	.16	14-16	.17		
3-13	.2	7-25	.12	14-17	.09		
3-14	.32	8-9	.14	14-18	.2		
3-15	.06	8-10	.23	14-19	.07		
3-16	.12	8-11	.19	14-20	.06		
3-17	.24	8-12	.04	14-21	.21		
3-18	.31	8-13	.03	14-22	.26		
3-19	.27	8-14	.4	14-23	.29		
3-20	.14	8-15	.09	14-24	.15		
3-21	.16	8-16	.05	14-25	.34		
3-22	.13	8-17	.15	15-16	.2		
3-23	.26	8-18	.38	15-17	.22		
3-24	.4	8-19	.42	15-18	.17		
3-25	.32	8-20	.16	15-19	.05		
4-5	.16	8-21	.07	15-20	.41		
4-6	.29	8-22	.18	15-21	.52		
4-7	.17	8-23	.2	15-22	.19		
4-8	.09	8-24	.19	15-23	.26		
4-9	.24	8-25	.08	15-24	.13		
4-10	.31	9-10	.04	15-25	.03		
4-11	.27	9-11	.16	16-17	.09		
4-12	.14	9-12	.29	16-18	.16		
4-13	.16	9-13	.35	16-19	.25		
4-14	.13	9-14	.17	16-20	.09		
4-15	.26	9-15	.16	16-21	.11		
4-16	.29	9-16	.38	16-22	.21		
4-17	.34	9-17	.42	16-23	.03		
4-18	.21	9-18	.16	16-24	.16		
4-19	.08	9-19	.07	15-25	.42		
4-20	.04	9-20	.08	17-18	.12		
4-21	.16	9-21	.21	17-19	.06		
4-22	.29	9-22	.29	17-20	.14		
4-23	.38	9-23	.41	17-21	.08		
4-24	.26	9-24	.5	17-22	.36		
4-25	.09	9-25	.02	17-23	.48		

Table 13 shows the values recorded for Adjusted R-Squared, MMRE and Pred (25%) for each of the data set using the methodology detailed before. Looking at the values, we can clearly say that the methodology performed exceedingly well. The performance of the proposed methodology is further explained with respect to each data set below.

Table 13: Adjusted R – Squared, MMRE and Pred Values for the Methodology on each Data Set

	Adj R^2	**MMRE**	**Pred(25%)**
DS 1	.85	30%	65%
DS 2	.8	28%	57%
DS 3	.92	23%	75%
DS 4	.79	29%	74%
DS 5	.68	43%	58%
DS 6	.75	42%	52%

For DS 1 the methodology resulted in Adj R^2 = .85 and MMRE = 30% which shows that the model built was good and also the bias was low. Also the Pred value was high. The methodology performed similarly on DS 2. Though there was a slight reduction in Adj R^2 = .8 value and Pred = 57% value, MMRE = 28% reduced. The methodology performed exceptionally well (Adj R^2 = .92, MMRE = 23% and Pred (25%) = 75%) for DS 3 as the missing mechanism was MCAR. For DS 4, the performance was similar to that of DS 1 and DS 2 except for an increase in Pred values (Adj R^2 = .79, MMRE = 29% and Pred (25%) = 74%). The methodology gave the lowest performance for DS 5 (Adj R^2 = .68, MMRE = 43% and Pred (25%) = 58%). A reasonable model was built but the bias was a little high. The reason could be because of high percentage of data missing. Again for DS 6, it performed (Adj R^2 = .75, MMRE = 42% and Pred (25%) = 52%) pretty well.

Overall, the performance of the methodology was satisfactory. It built models with an average Adjusted R-Squared value of .798 which is almost .8, had an average MMRE of 33% and an average Pred (25%) value of 64%. An average value of .8 indicates that all the models built from the data sets imputed using the methodology are considerably good and having an average MMRE of 33% shows the estimates had less bias. Moreover, in every model built there were on an average 64% of cases having relative error less than or equal to 25%. The above statistics suggest that the methodology could be used for different kinds of data sets (such as small, medium, and large) and

under different conditions (such as pattern of missing data, % of missing data and under different missing mechanisms). Figure 13 represents the performance of the methodology in terms of both goodness of fit and accuracy.

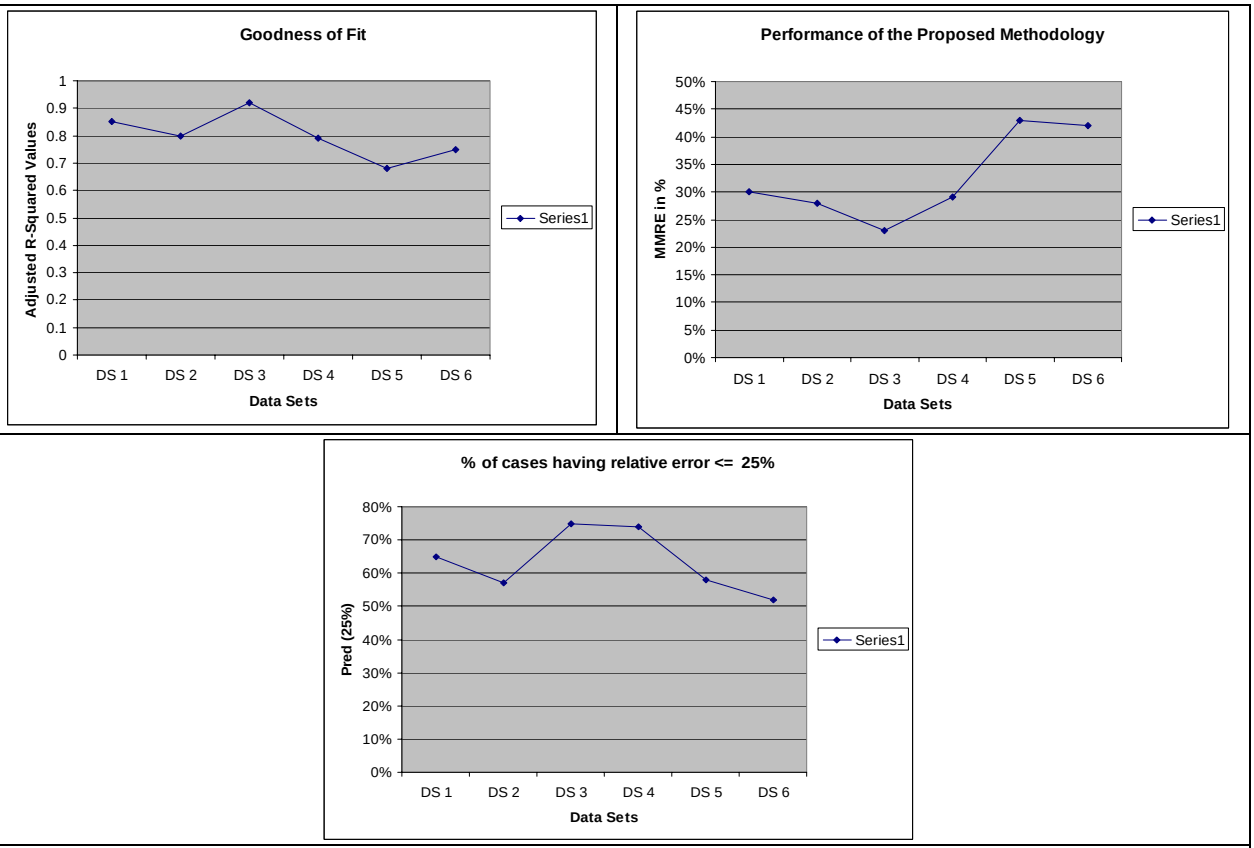


Fig 13: Graphical Representation of Adjusted R-Squared, MMRE & Pred Values for the Methodology on each Dataset

CHAPTER 7 COMPARISON OF RESULTS

In this chapter, we compare and evaluate the performance of the proposed methodology with respect to each method implemented in chapter 5.

7.1 Methodology Vs LD

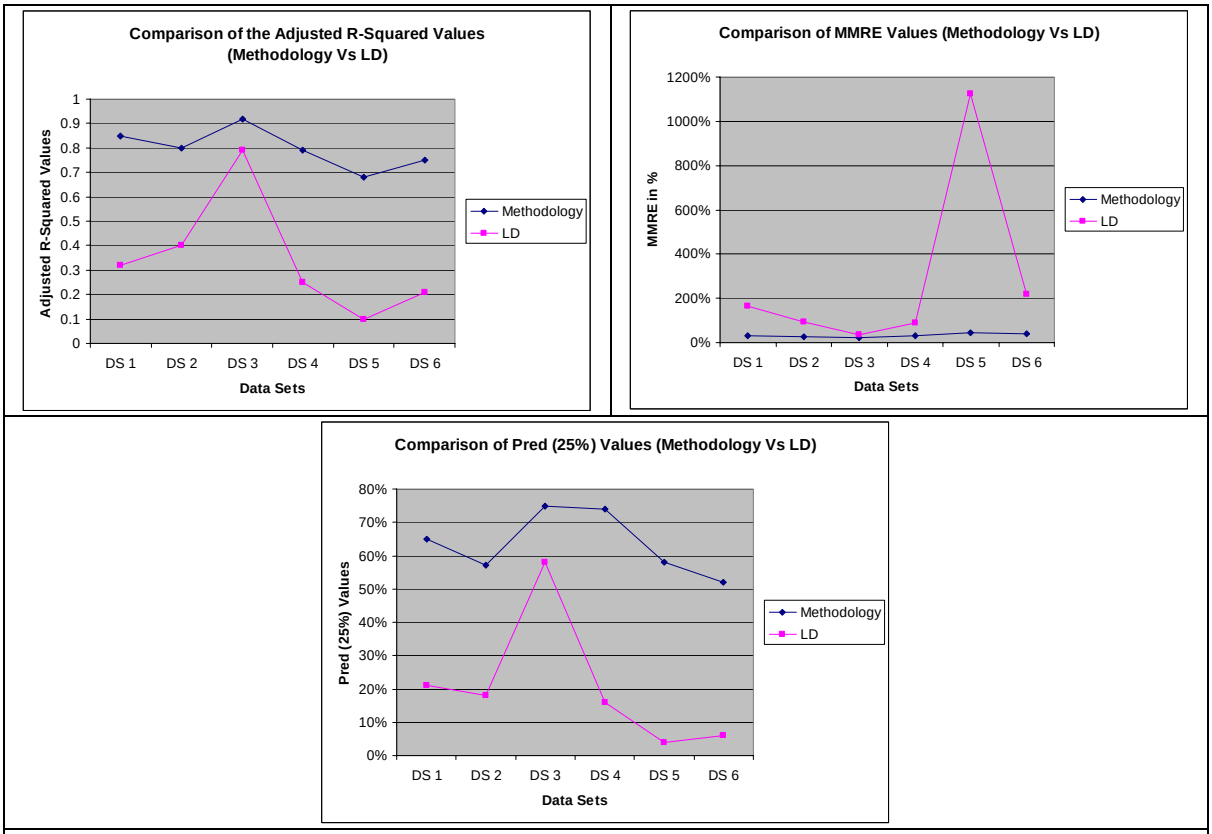


Fig 14: Comparison of Adjusted R – Squared, MMRE and Pred Values for the Methodology Vs LD

The proposed methodology outperformed LD with respect to each of the six real-time data sets. It built models with an average Adjusted R-Squared value of .798 and LD built models with an average Adjusted R-Squared value of .34. We can clearly see from the first graph that LD was not even close for DS 1, DS 2, DS 4, DS 5 and DS 6 in terms of goodness of fit except for DS 3. For DS 3, LD performed reasonably well as the data were missing completely at random. The MMRE values from the second graph show us

the irregular performance of LD. The average MMRE value for LD was 288% when compared with the methodology's average MMRE value of 33%. LD performed the worst for DS 5 as the percentage of data missing was high. The values clearly suggest that the estimates drawn from the data set imputed using LD were highly biased. Also, the average Pred value using LD was 21% where in the average Pred values for the methodology was 64%. A small percentage (21%) of cases had relative error less than or equal to 25% when estimates were drawn using LD. The overall performance of LD was far below that of the methodology.

7.2 Methodology Vs MI

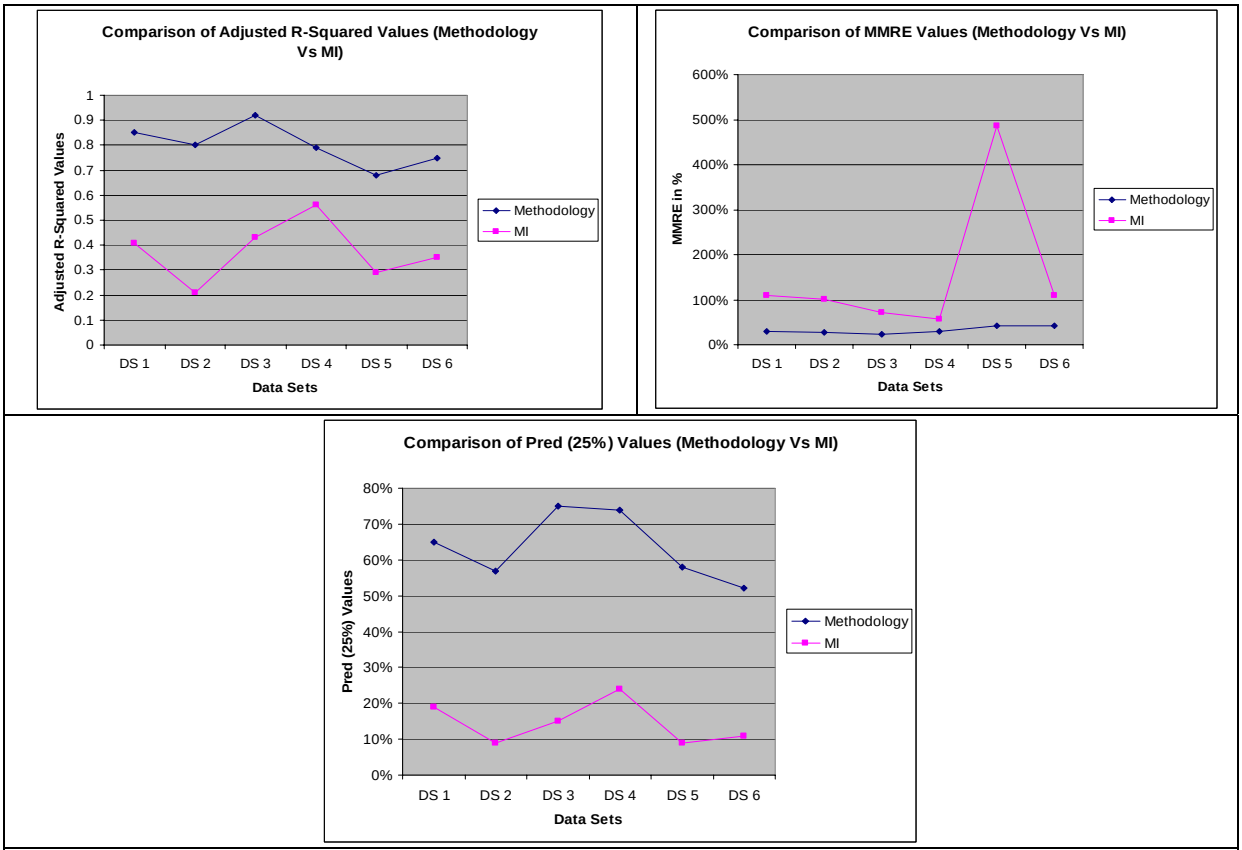


Fig 15: Comparison of Adjusted R – Squared, MMRE and Pred Values for the Methodology Vs MI

The proposed methodology outperformed MI also. The average Adjusted R-Squared value for MI was .37, which is just a little better than that of LD. Other than DS 5, MI did not exhibit high irregular behavior. But the irregular behavior with respect to DS 5 may be again attributed to the high percentage of missing data. However, MI was not as erroneous as LD. The average MMRE for MI was 156%, which is considerably high when compared to the average MMRE of the methodology that is 33%. The estimates drawn using MI were highly biased too. Also the average Pred value was 15%, which is lower than LD. Overall, MI showed a low performance when compared to the proposed methodology.

7.3 Methodology Vs Sequential Hot-Deck

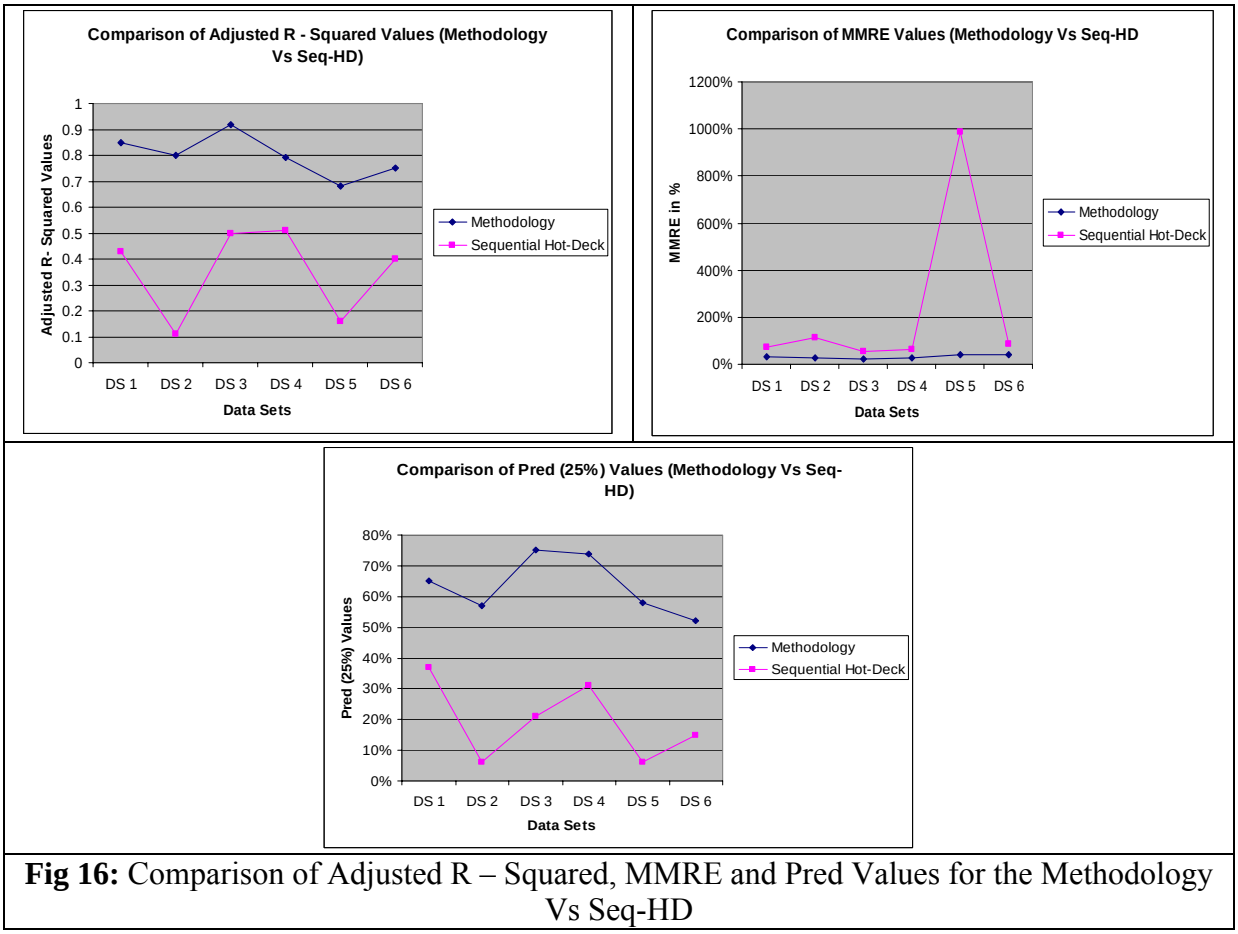


Fig 16: Comparison of Adjusted R – Squared, MMRE and Pred Values for the Methodology Vs Seq-HD

The average Adjusted R-Squared value for Sequential HD was .35 and the average MMRE value was 230%. The proposed methodology again outperformed on all the six data sets when compared to Sequential HD. The models built were of poor quality and the estimates were highly biased. Sequential HD too had a high MMRE like LD for DS 5. The average percentage of cases having relative error less than or equal to 25% was 19% for Sequential HD. The proposed methodology had an average Pred value of 64%. The proposed methodology can be rated well above Sequential HD, which is evident from the graphs noted in fig 16.

7.4 Methodology Vs Random Hot-Deck

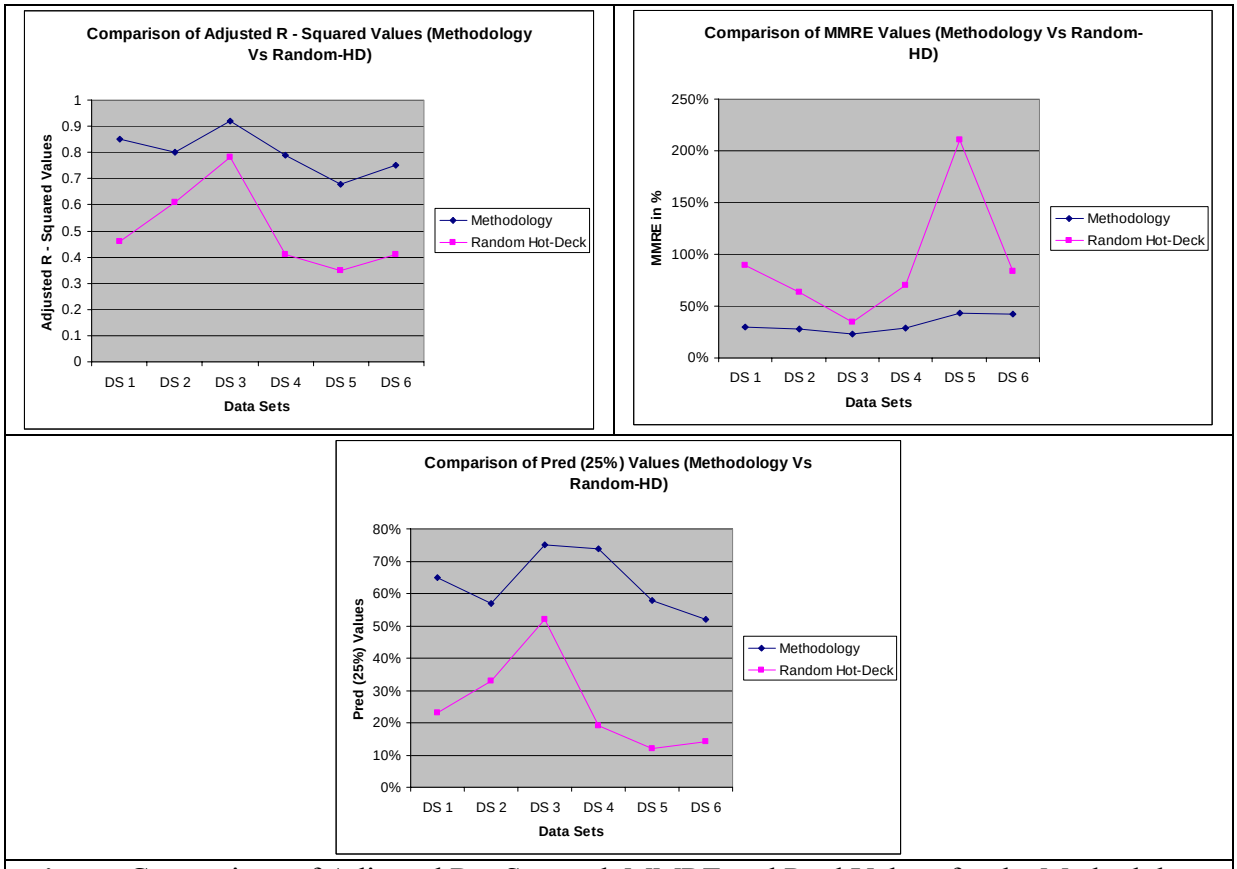


Fig 17: Comparison of Adjusted R – Squared, MMRE and Pred Values for the Methodology Vs Random-HD

Random HD built models with an average Adjusted R-Squared value of .5, which is the best so far from the methods implemented in chapter 5. But it is still below par when compared to the proposed methodology's value of .798. For DS 3 Random HD had performed the best with values comparatively closer to that of the proposed methodology. But again for DS 5, Random HD had a high MMRE of 211%. The average MMRE for Random HD is much better when calculated for the other five data sets (excluding DS 5).

The average MMRE value for Random HD is 92% which is however almost thrice that of the proposed methodology (33%). However, the average Pred value for Random HD was 26%, which is much better when compared to the previous methods. On the whole, the performance of the proposed methodology was much better in terms of both goodness of fit and prediction accuracies which can be visualized through the fig 17 above.

7.5 Methodology Vs SRPI

The average Adjusted R-Squared value for SRPI was .59, which is reasonable, but still considerably lower than the average value of the proposed methodology (.798). But the average MMRE value, which is 63%, improved considerably than the previous methods. For DS 3, SRPI built a model almost similar to the one built using the proposed methodology. SRPI performed pretty well for DS 1 and DS 2. But it under performed for DS 4, DS 5 and DS 6. Its performance particularly degraded for DS 5 and DS 6. On the whole, the average Pred value of SRPI was 35%, which is still nearly half the average value of the proposed methodology (64%). The performance of SRPI was better than the previous methods and did considerably well on a few data sets but could not match the performance of the proposed methodology which can be noticed from fig 18.

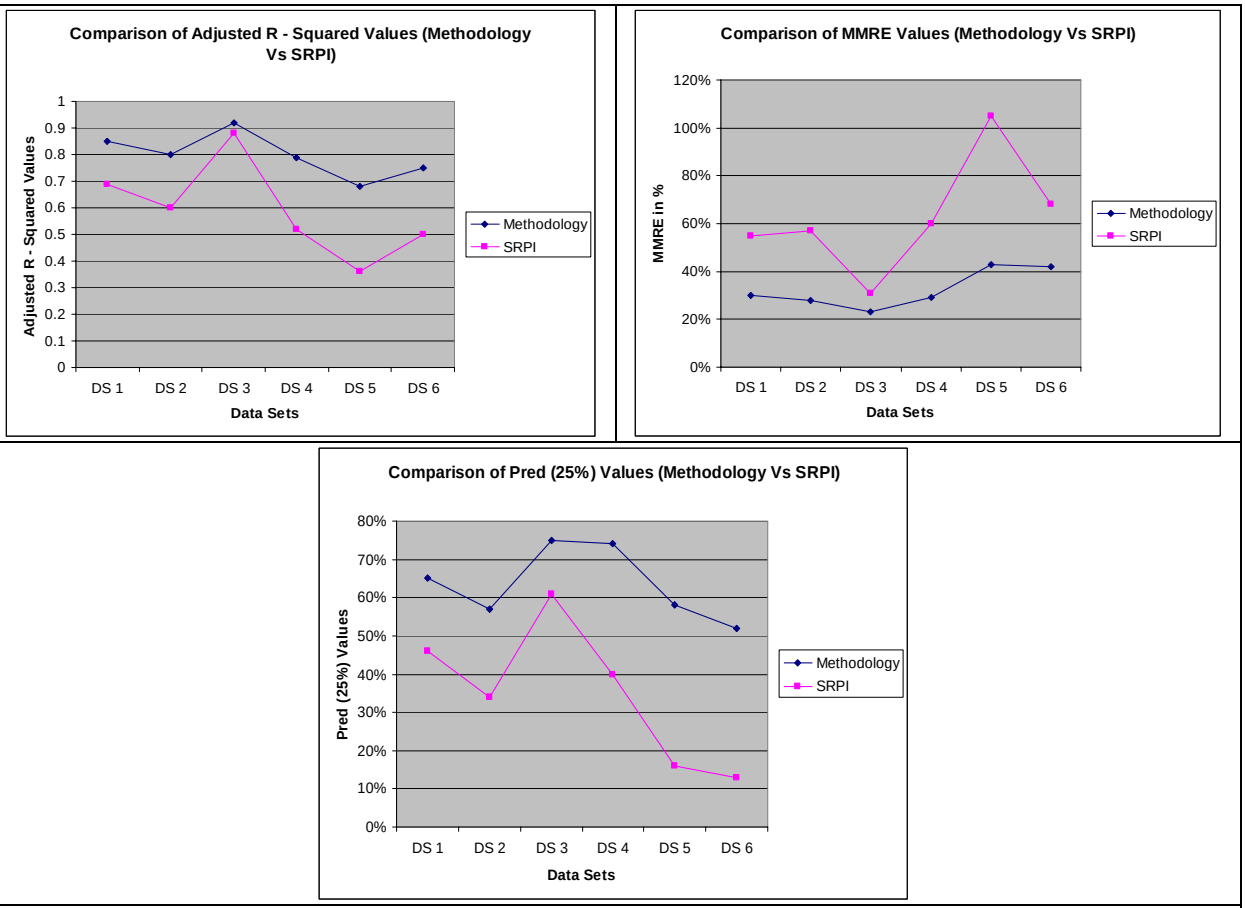


Fig 18: Comparison of Adjusted R – Squared, MMRE and Pred Values for the Methodology Vs SRPI

7.6 Methodology Vs *k*-Nearest Neighborhood Approaches (Euclidean Distance)

The average Adjusted R-Squared value for the Euclidean Distance Method was .64 which is a decent fit but still less than the average value of the proposed methodology (.798). For DS 3, it built a model which is very much similar to the one built by the proposed methodology. It performed similar to SRPI with an average MMRE value of 59% (a little less than SRPI) and an average Pred value of 39%. The performance was reasonable with respect to the first two data sets but degraded highly for the last two (like SRPI). The overall performance of the proposed methodology was better than that of the Euclidean Distance Method.

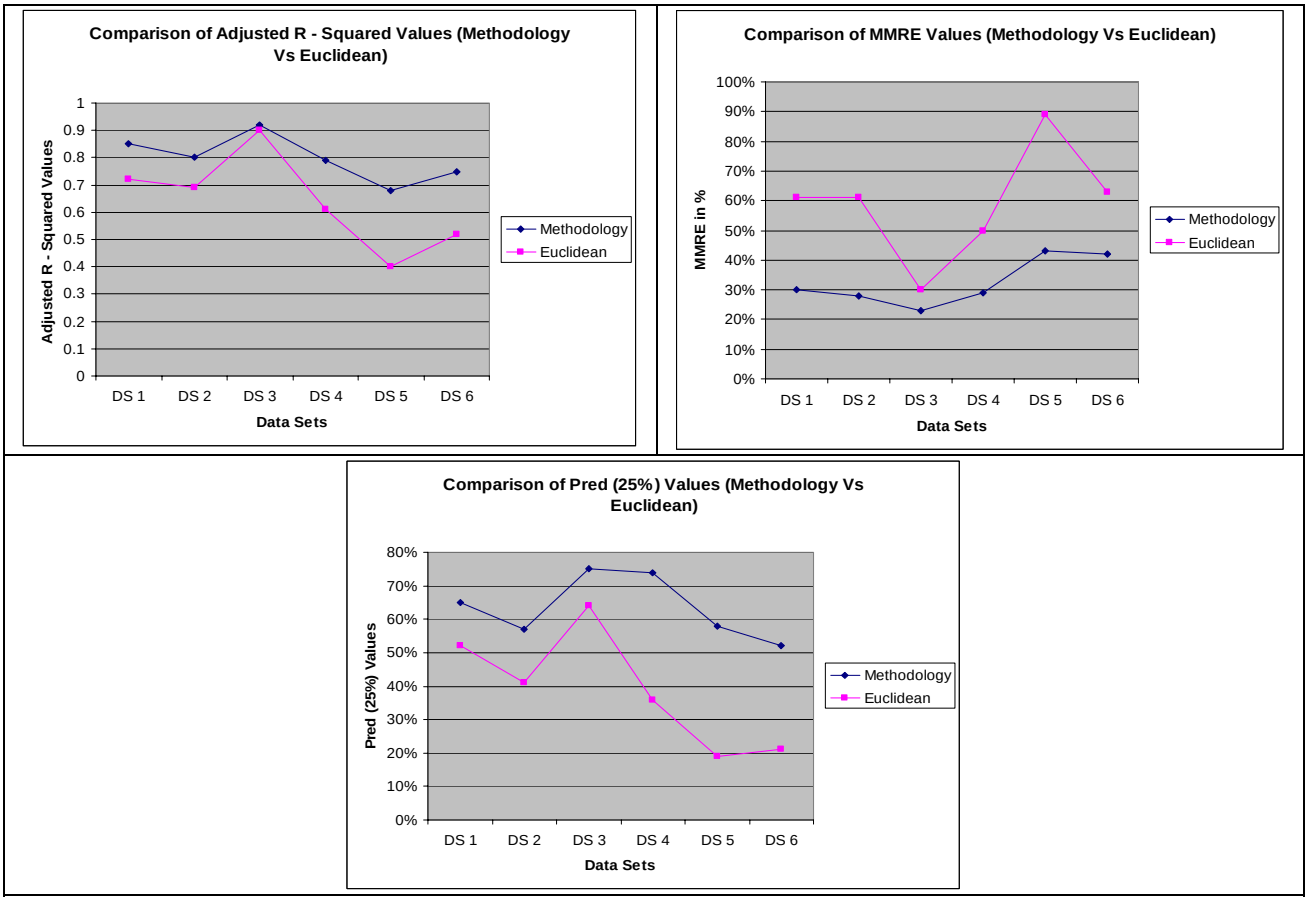


Fig 19: Comparison of Adjusted R – Squared, MMRE and Pred Values for the Methodology Vs Euclidean

7.7 Methodology Vs *k*-Nearest Neighborhood Approaches (Manhattan Distance)

The performance of the Manhattan Distance Method was again similar to that of the Euclidean Distance Method. It has an average Adjusted R-Squared value of .69 (slightly higher than Euclidean Distance Method). The method built models reasonably well and close to the ones built by the proposed methodology for data sets 1 to 4. It built a model almost similar to the one constructed using the proposed methodology for DS 3. Its performance degraded for DS 5 and DS 6. The reasons would have to be the high percentage of missing data and NI conditions respectively. The average MMRE value was 57% and an average Pred value was 38%, which are comparatively low to the

average MMRE (33%) and Pred values (64%) of the proposed methodology. The overall performance though was still low when compared to that of the proposed methodology.

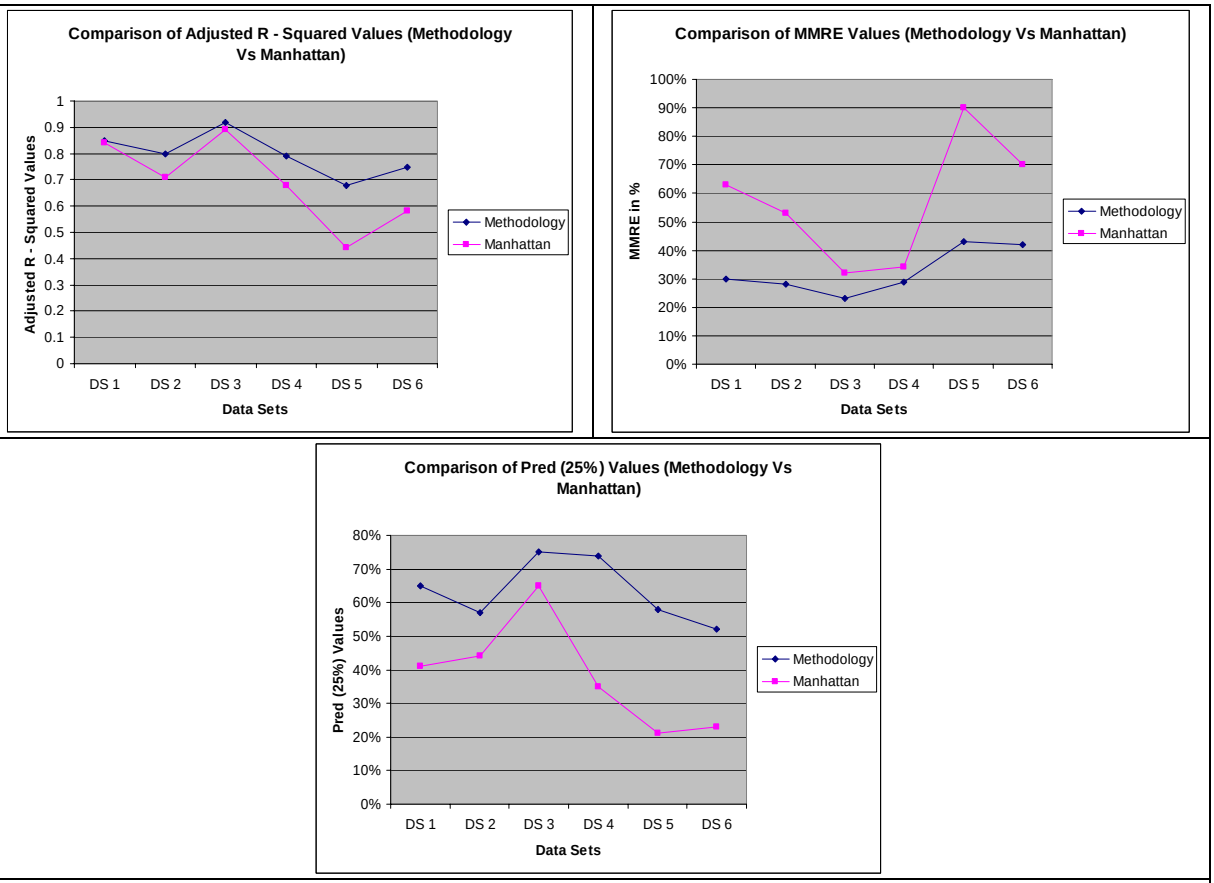


Fig 20: Comparison of Adjusted R – Squared, MMRE and Pred Values for the Methodology Vs Manhattan

7.8 Methodology Vs *k*-Nearest Neighborhood Approaches (Maholanobis Distance)

The performance of the Maholanobis Distance Method was more or less similar to the previous two distance methods. It has an average Adjusted R-Squared value of .6. The method built models reasonably well and close to the ones built by the proposed methodology for data sets 1 to 3. Its performance degraded for DS 4, DS 5 and DS 6. The average MMRE value was 60% and an average Pred value was 38%, which are comparatively low to the average MMRE (33%) and Pred values (64%) of the proposed

methodology. The overall performance still did not improve when compared to that of the proposed methodology.

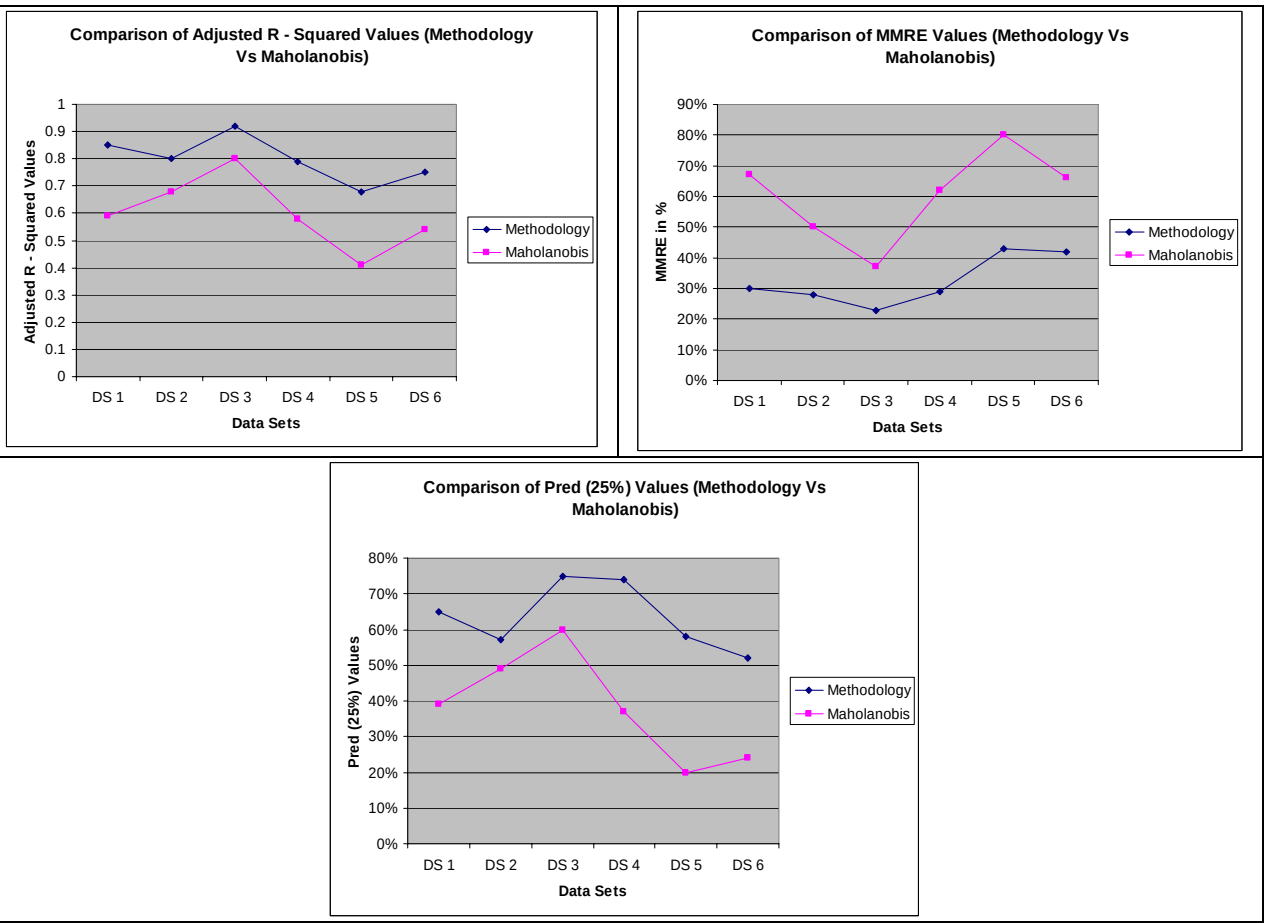


Fig 21: Comparison of Adjusted R – Squared, MMRE and Pred Values for the Methodology Vs Maholanobis

7.9 Methodology Vs *k*-Nearest Neighborhood Approaches (Correlation Distance)

The performance of the Correlation Distance Method was more or less similar to the previous distance methods. It has an average Adjusted R-Squared value of .6 (similar to Maholanobis Distance Method). The method built models reasonably well and close to the ones built by the proposed methodology for data sets 1 to 3. It built a model almost similar to the one constructed using the proposed methodology for DS 3. Its performance

degraded for DS 4, DS 5 and DS 6. The average MMRE value was 60% (similar to Maholanobis Distance Method) and an average Pred value was 42%, which are still comparatively low to the average MMRE (33%) and Pred values (64%) of the proposed methodology. None of the distance methods outperformed the proposed methodology so far.

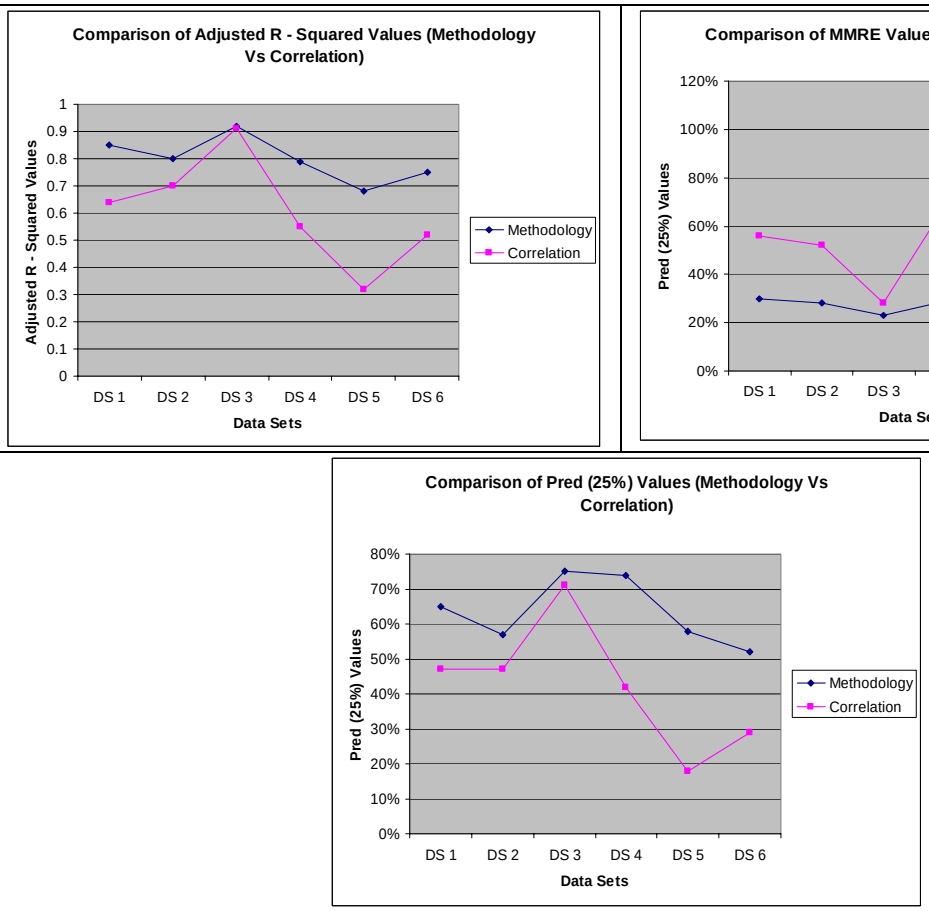


Fig 22: Comparison of Adjusted R – Squared, MMRE and Pred Values for the Methodology Vs Correlation

7.10 Methodology Vs *k*-Nearest Neighborhood Approaches (Cosine Distance)

The performance of the Cosine Distance Method was also more or less similar to the previous distance methods with an average Adjusted R-Squared value of .55 (lesser than two previous methods). It built models reasonably well for data sets 1 to 4. It built a

model comparatively close to the one constructed using the proposed methodology for DS 3. Its performance degraded for DS 5 and DS 6. The average MMRE value was 63% (similar to two previous methods) and an average Pred value was 42%, which are comparatively low to the average MMRE (33%) and Pred values (64%) of the proposed methodology. The performance of the proposed methodology still was the best.

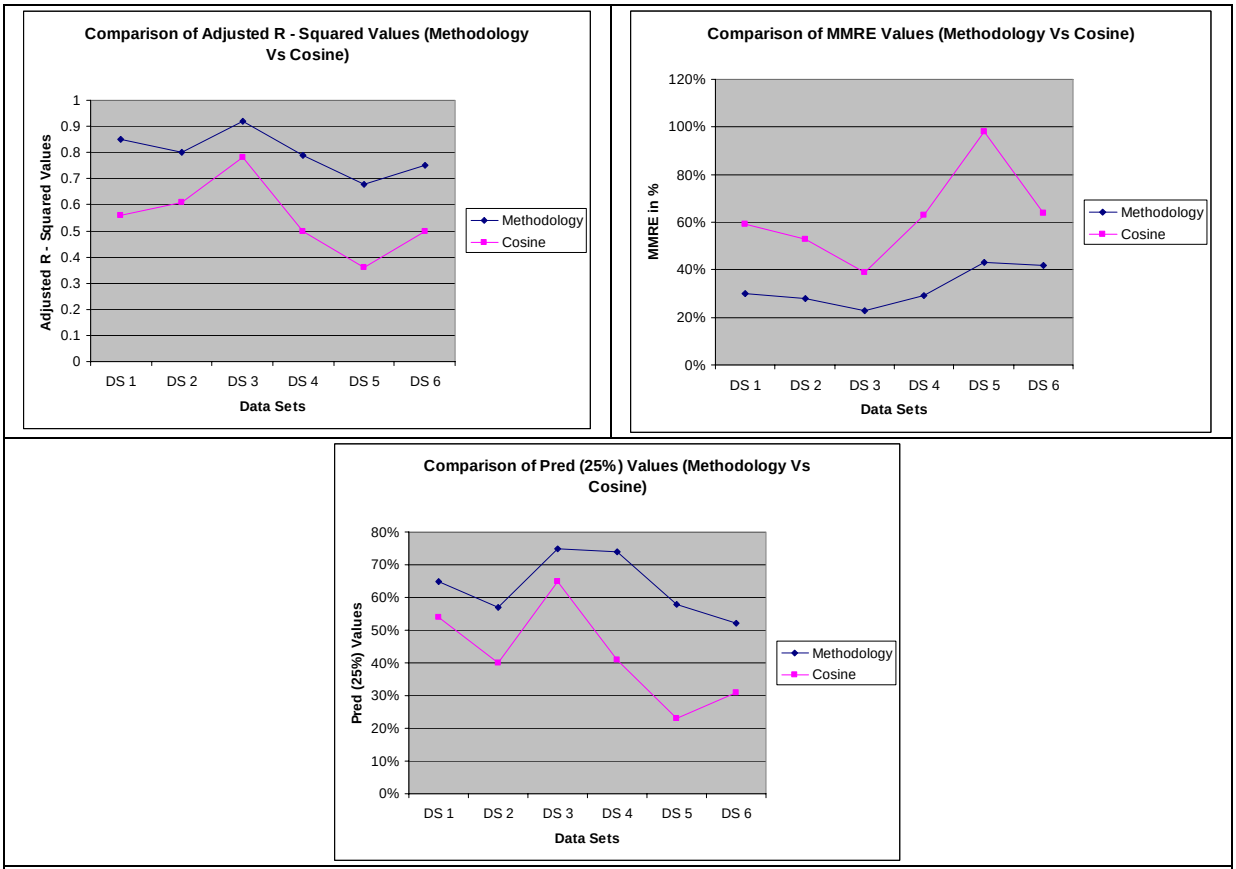


Fig 23: Comparison of Adjusted R – Squared, MMRE and Pred Values for the Methodology Vs Cosine

7.11 Methodology Vs *k*-Nearest Neighborhood Approaches (Squared-Chord Distance)

Squared Chord Distance Method did not perform better than the previous distance methods. It had an average Adjusted R-Squared value of .61. It built models reasonably well for data sets 1 to 3. It built a model almost similar to the one constructed using the

proposed methodology for DS 3. Its performance degraded for DS4, DS 5 and DS 6. The average MMRE value was 65% and an average Pred value was 39%, which are still low than the average MMRE (33%) and Pred values (64%) of the proposed methodology.

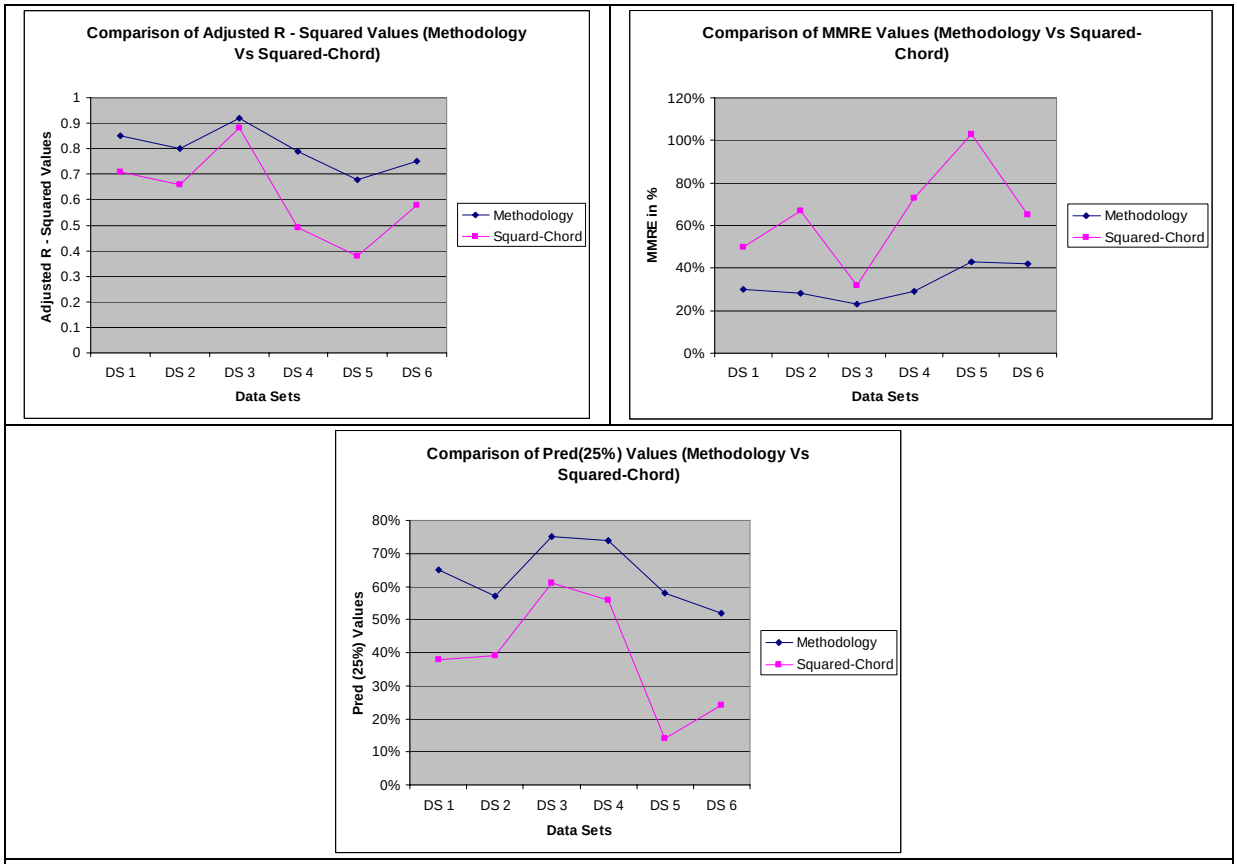


Fig 24: Comparison of Adjusted R – Squared, MMRE and Pred Values for the Methodology Vs Squared-Chord

7.12 Methodology Vs *k*-Nearest Neighborhood Approaches (Combination Method)

The performance of the Combination Method was much better than all the previous methods with an average Adjusted R-Squared value of .68. It built models reasonably well for data sets 1 to 4. It built models similar to the ones constructed using the proposed methodology for DS1, DS 2, and DS 3. Its performance though degraded for DS 5 and DS 6. The average MMRE value was 48% (which is much lesser than all the previous methods) and an average Pred value was 50% (which is higher than all the

previous methods). The performance of the Combination Method is the closest to the performance of the proposed methodology so far. There was a significant improvement in the performance of the Combination method. The estimates were not highly biased as with the other previous methods and the Pred value shows the half the cases had relative error less than or equal to 25% which is a reasonable performance.

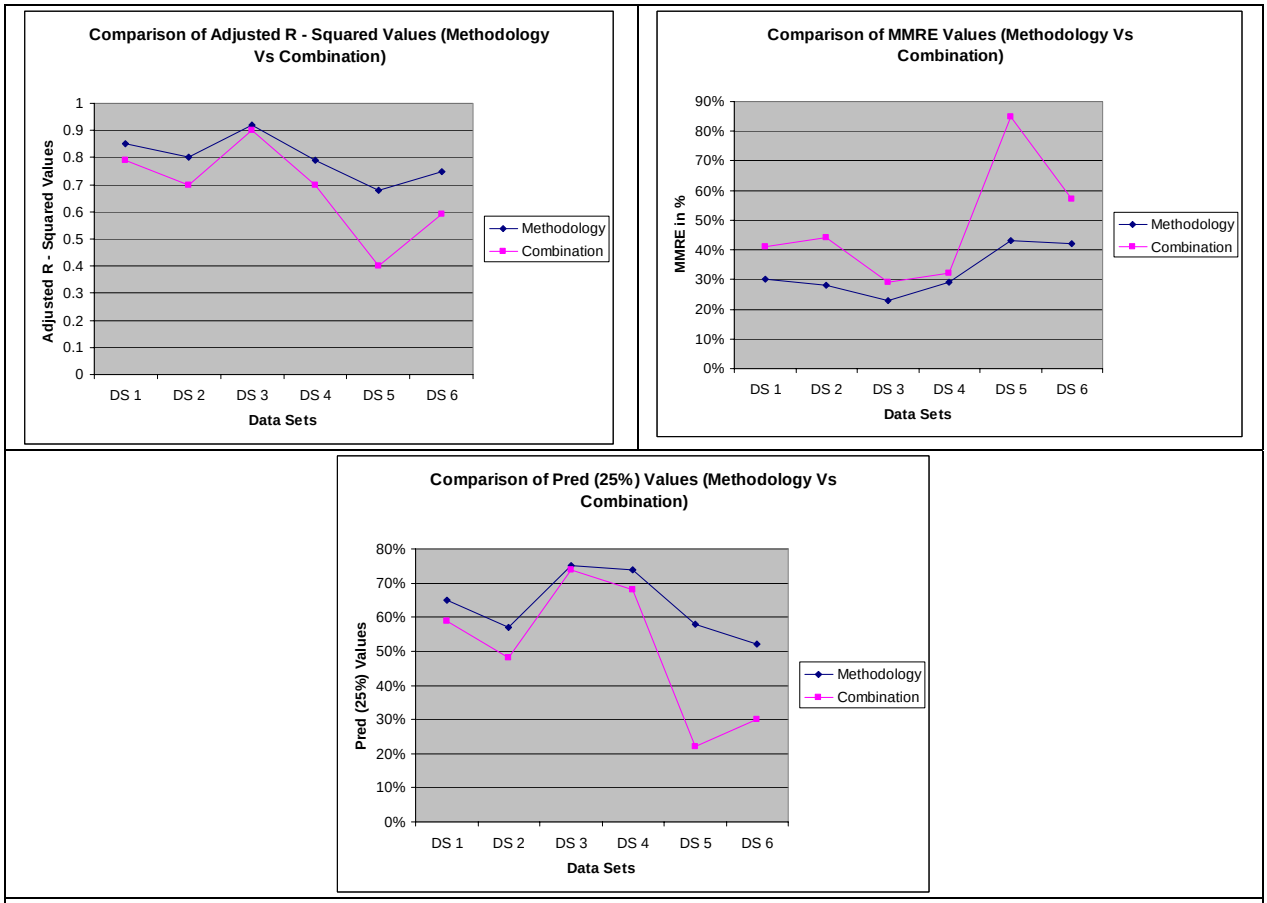


Fig 25: Comparison of Adjusted R – Squared, MMRE and Pred Values for the Methodology Vs Combination

7.13 Methodology Vs *k*-Nearest Neighborhood Approaches (FIML)

FIML's performance was the best of all the methods implemented in chapter 5. It had an average Adjusted R-Squared value of .69. It built models reasonably well for all the data sets. It built models similar to the ones constructed using the proposed

methodology for DS1, DS 2, DS 3, and DS 6. Its performance was a little low for DS 4 and DS 5 though. The average MMRE value was 43% (lesser than all the previous methods) and an average Pred value was 57% (higher than all the previous methods). FIML's performance was the closest to the performance of the proposed methodology. The estimates were not biased as with the other previous methods and did not show any irregular behavior particularly with DS 5 (where the percentage of data missing was high). The Pred value shows the more than half the cases had relative error less than or equal to 25% which is a reasonable performance. For DS 6 the Pred value using FIML was better than the Pred value of the proposed methodology. FIML performed better than most of the previous methods and was the closest to the proposed methodology.

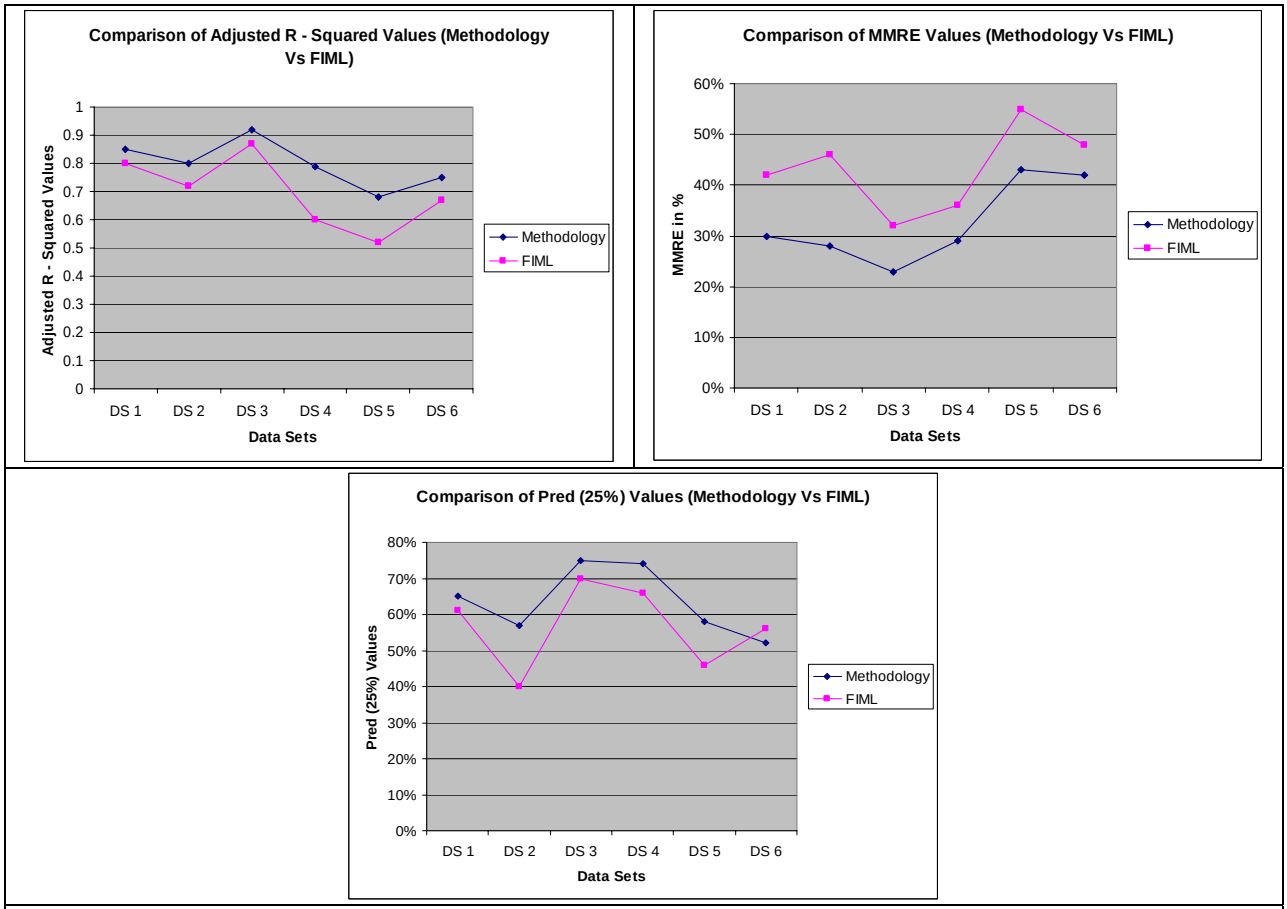


Fig 26: Comparison of Adjusted R – Squared, MMRE and Pred Values for the Methodology Vs FIML

7.14 Overall Analysis

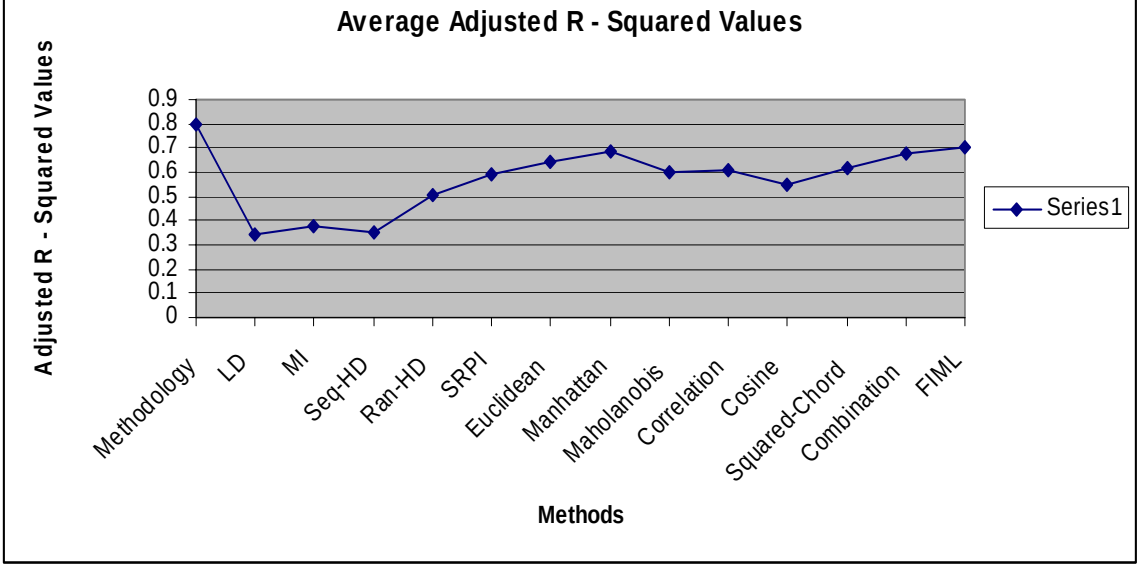
The overall performance of all the methods with respect to all the six data sets is depicted in the graphs below. The first graph represents the average goodness of fit (Adjusted R – Squared values) achieved by each of the methods when implemented on all the six data sets. The second graph represents the average MMRE values and the third graph represents the average Pred (25%) values. The same values have been listed below in table 14.

Table 14: Overall performance of each method w.r.t the six data sets

	Avg (Adj. R - Squared)	**Avg (MMRE)**	**Avg (Pred (25%))**
Methodology	**0.798**	**33%**	**64%**
LD	0.345	288%	21%
MI	0.375	156%	15%
Seq-HD	0.351	230%	19%
Ran-HD	0.503	92%	26%
SRPI	0.592	63%	35%
Euclidean	0.64	59%	39%
Manhattan	0.69	57%	38%
Maholanobis	0.6	60%	38%
Correlation	0.606	60%	42%
Cosine	0.551	63%	42%
Squared-Chord	0.616	65%	39%
Combination	0.68	48%	50%
FIML	0.696	43%	57%

By looking at the first graph in fig 27 it can be clearly said that the proposed methodology built the best models with an average Adjusted R – Squared value = .798. FIML (Adjusted R – Squared value = .696) and Manhattan Distance Method (Adjusted R – Squared value = .69) Combination Method (Adjusted R – Squared value = .68) were the only methods whose models could be compared with the ones built by the proposed methodology. SRPI and the remaining distance methods built models more or less of the

same quality. LD, MI, and Sequential HD built models of poor quality. Random HD built models of average quality. The proposed methodology had the best overall performance in building good quality models.



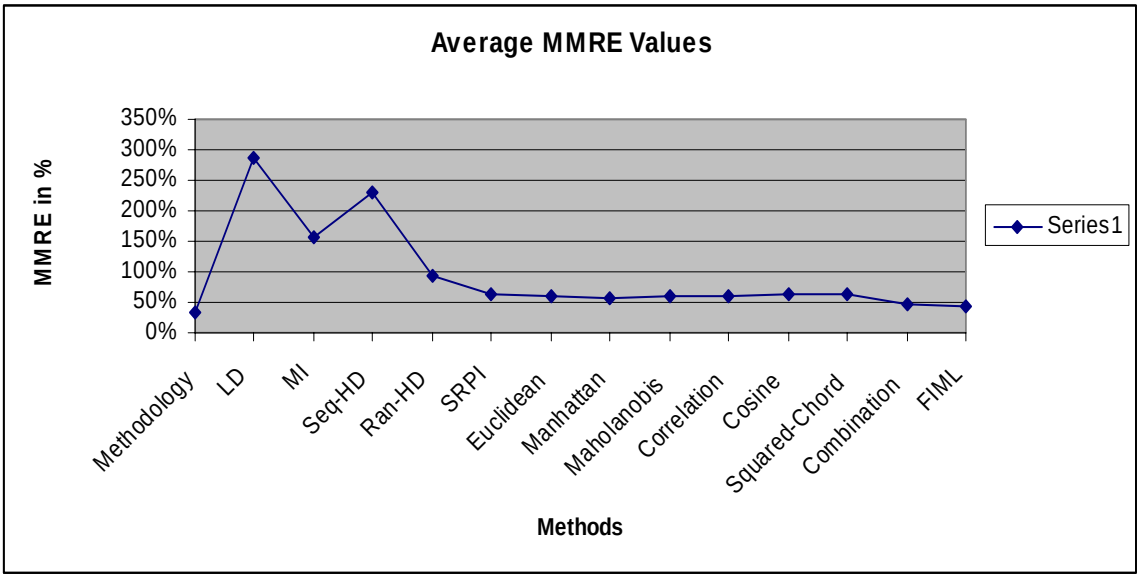
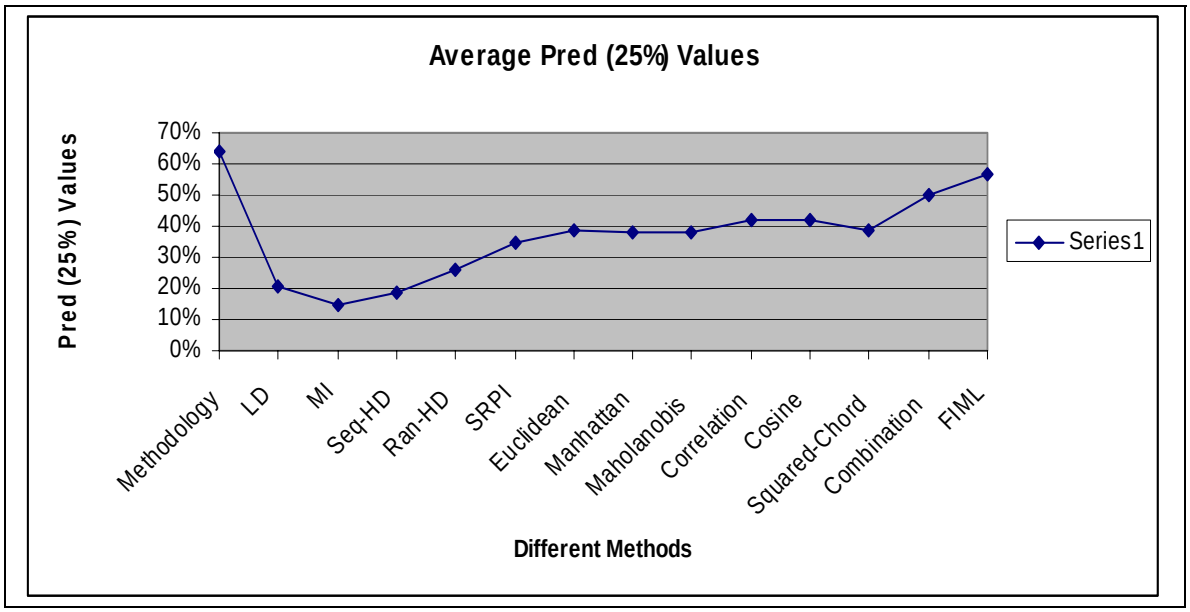


Fig 27: Average Adjusted R-Squared Values, MMRE, and Pred Values for all the Methods

(fig 27 cont.)



The average MMRE value for the proposed methodology was 33%. Other than FIML (MMRE = 43%) and Combination Method (MMRE = 48%), none of them even came close to the average value of the proposed methodology. All the estimates drawn from the data sets formed by the other methods were biased. LD, MI and Sequential HD gave the highest biased estimates. Random HD also gave highly biased estimates but not as much as LD, MI and Sequential HD.

The other distance methods performed on an average around (MMRE = (55%-65%)) which is considerate when compared with LD, MI, Sequential HD or Random HD. The best estimates were drawn from the data sets imputed using the proposed methodology. Again, FIML (Pred (25%) = 57%) and Combination Method (Pred (25%) = 50%) had Pred values close to the proposed methodology (Pred (25%) = 64%). LD, MI, Sequential HD and Random HD had Pred values less than 30%. More than 70% of the cases had relative error greater than 25%. All the other distance methods had Pred values between (35% -42%). The proposed methodology had 64% of cases, which had relative error less than 25%. None of the methods out performed the methodology.

CHAPTER 8 FINAL DISCUSSIONS

In this study, our aim was to investigate if prediction accuracies improved when completeness of a data set is enhanced using imputation techniques. We tried to maximize the response in the data set for the same. We compared the performance of four different imputation strategies (LD, MI, 10 variants of HD, and FIML) on enhancing completeness in incomplete software project data sets. We evaluated their impact by implementing them on six different real-time software project data sets, which are classified into different categories based on their inherent properties. The reliability of the constructed data sets using these techniques was further tested by building prediction models using stepwise regression.

Furthermore, we implemented a hybrid methodology to overcome the limitations of most imputation methods. The methodology initially runs a clustering algorithm on each of the data sets and forms homogenous clusters. Next, it selects the appropriate cluster as well as the donor(s) from the cluster for each missing case and thus imputes data. In this chapter, we discuss our findings. We also highlight the conditions to be considered and measures to be taken while using an imputation technique. We note the ideal and worst conditions for each method. We finally detail on the appropriateness of each method for data imputation.

Observations and recommendations based from the experimental results:

- After reviewing the results, we can say that all the methods performed better than LD. Only in 2 instances LD performed better than MI and Sequential HD. In both these instances MI and Sequential HD did not perform well because of the pattern in which data were missing. Also, whenever the data set had few complete cases, LD

underperformed (DS1, DS4, & DS5). When missing data are not confined to a small percentage of cases, LD performed badly. The performance of LD deteriorates as the number of cases with missing values increase. Also, LD underperformed when the missing mechanisms were MAR (DS2) and NI (DS6).

- We suggest not to use LD when the missing pattern is univariate or monotonous. LD is highly unreliable when data are missing in high percentage. We also suggest researchers use LD only when the missing mechanism is MCAR and when the missing percentage is < 5%.
- MI and Sequential HD did not perform well when the missing patterns were monotonous (DS2) and univariate (DS3). The reason is obvious. The same value/donor was used to impute the missing values in both cases thus distorting the underlying distribution.
- Pattern in which the data are missing plays an important while using these methods. Even when the pattern is arbitrary, these methods may not perform well if less number of variables contributes towards large number of missing values.
- Moreover, we found that MI and Sequential HD may not be least biased under MAR or NI conditions (DS1 & DS6). Random HD performed slightly better than Sequential HD in most cases but did not yield reasonable fits.
- We suggest not using MI or Sequential HD when missing pattern is univariate and suggest using MI, Sequential HD or Random HD only under MCAR conditions and when the percentage of missing data is between 5%-10%.
- Though Random HD may be used in MAR conditions, the percentage of missing data should be less than 10%.

- SRPI along with other *k*-NN HD methods performed more or less the same. Overall, the Manhattan Distance Metric and the Combination Method yielded the best results among all of them.
- Both of them outperformed FIML in a few instances (DS3 & DS4). It may be due to the reason that HD variants particularly work well with smaller sized data sets.
- All the methods performed well under MCAR and MAR conditions but yielded biased results under NI conditions (DS6). Their performance did not rely on the size of the data set or the missing pattern. We recommend using HD variants (particularly Manhattan and Combination Methods) when the data sets are relatively small/medium and the missing mechanism is not NI.
- FIML performed more or less similar to Manhattan Distance Metric and the Combination Methods in all instances excepting the one under NI conditions (DS6). FIML gave least biased estimates under NI conditions.
- FIML works well for larger data sets and even under NI conditions. Though it may be computationally demanding, we recommend using FIML under NI conditions in particular.
- None of the methods including FIML performed even reasonably well when high percentage of data was missing (DS5). FIML may perform reasonably in such situations but we are not thoroughly convinced.
- In general, the performance of all techniques degraded as the missing percentage increases. We recommend not imputing when the data set has missing percentage above 50. Imputation should be used only when necessary but not to make the data set look good by making it complete.

- The proposed methodology was designed by taking into aspect the missing mechanism, data set size, missing percentage and the pattern in which the data are missing. The idea was to overcome these limitations faced by traditional imputation methods.
- To begin with, the methodology is not computationally demanding and can work extremely well with all sizes of data sets. As it identifies "like" cases and clusters them, before choosing a donor, there is a high reliability that missing cases are often imputed with the "most probable" values.
- Due to the very nature of the method to form homogenous clusters, the missing pattern or the missing mechanism cause no degradation in its performance.
- However, the only factor that can influence its performance is the percentage of missing data. The performance of the methodology may degrade when high percentage of data is missing. It can be said that the estimates provided by the methodology for data sets having missing percentage > 45% may be biased.
- In our study though, we ended up with credible data sets and reasonable models were built when 46% of data were missing (DS 5).
- However, more number of data sets needs to be tested before conforming the performance of the methodology when missingness is present in high quantities (> 40%).
- The proposed methodology can be used under MCAR, MAR or NI conditions, can be used for data set of any size, and for any kind of missing pattern or a combination of all these characteristics. Finally, we suggest the following methods (based on our classification schema) to be used for data sets having different characteristics.

Table 15: Recommendation of Imputation Method for Each Class in a Small Sized Data Set

Small	MCAR	< 15%	Univariate	LD, MI or any simple data imputation should work fine
			Monotonous	
			Arbitrary	
		> 15% & < 30%	Univariate	LD, MI or any simple data imputation should work fine
			Monotonous	
			Arbitrary	
		> 30% & < 45%	Univariate	LD, MI or any simple data imputation should work fine
			Monotonous	
			Arbitrary	
		> 45%	Univariate	Do not impute
			Monotonous	
			Arbitrary	
	MAR	< 15%	Univariate	Any *k*-NN Method or Methodology
			Monotonous	Any *k*-NN Method or Methodology
			Arbitrary	MI, Any *k*-NN Method or Methodology
		> 15% & < 30%	Univariate	Any *k*-NN Method or Methodology
			Monotonous	Any *k*-NN Method or Methodology
			Arbitrary	Any *k*-NN Method or Methodology
		> 30% & < 45%	Univariate	Any *k*-NN Method or Methodology
			Monotonous	Any *k*-NN Method or Methodology
			Arbitrary	Any *k*-NN Method or Methodology
		> 45%	Univariate	Do not impute
			Monotonous	
			Arbitrary	
	NI	< 15%	Univariate	Any *k*-NN Method, Methodology, or FIML
			Monotonous	Any *k*-NN Method, Methodology, or FIML
			Arbitrary	Any *k*-NN Method, Methodology, or FIML
		> 15% & < 30%	Univariate	Methodology, or FIML
			Monotonous	Methodology, or FIML
			Arbitrary	Methodology, or FIML
		> 30% & < 45%	Univariate	Methodology, or FIML
			Monotonous	Methodology, or FIML
			Arbitrary	Methodology, or FIML
		> 45%	Univariate	Do not impute
			Monotonous	
			Arbitrary	

Table 16: Recommendation of Imputation Method for Each Class in a Medium Sized Data Set

Medium	MCAR	< 15%	Univariate	LD, MI or any simple data imputation should work fine
			Monotonous	
			Arbitrary	
		> 15% & < 30%	Univariate	LD, MI or any simple data imputation should work fine
			Monotonous	
			Arbitrary	
		> 30% & < 45%	Univariate	LD, MI or any simple data imputation should work fine but Hot-Deck Methods would be the best choice
			Monotonous	
			Arbitrary	
		> 45%	Univariate	Do not impute
			Monotonous	
			Arbitrary	
	MAR	< 15%	Univariate	Manhattan Distance Method, Combination Method or Methodology
			Monotonous	Manhattan Distance Method, Combination Method or Methodology
			Arbitrary	Manhattan Distance Method, Combination Method or Methodology
		> 15% & < 30%	Univariate	Manhattan Distance Method, Combination Method or Methodology
			Monotonous	Manhattan Distance Method, Combination Method or Methodology
			Arbitrary	Manhattan Distance Method, Combination Method or Methodology
		> 30% & < 45%	Univariate	Manhattan Distance Method, Combination Method or Methodology
			Monotonous	Manhattan Distance Method, Combination Method or Methodology
			Arbitrary	Manhattan Distance Method, Combination Method or Methodology
		> 45%	Univariate	Do not impute
			Monotonous	
			Arbitrary	
	NI	< 15%	Univariate	Methodology, or FIML
			Monotonous	Methodology, or FIML
			Arbitrary	Methodology, or FIML
		> 15% & < 30%	Univariate	Methodology, or FIML
			Monotonous	Methodology, or FIML
			Arbitrary	Methodology, or FIML
		> 30% & < 45%	Univariate	Methodology, or FIML
			Monotonous	Methodology, or FIML
			Arbitrary	Methodology, or FIML
		> 45%	Univariate	Do not impute
			Monotonous	
			Arbitrary	

Table 17: Recommendation of Imputation Method for Each Class in a Large Sized Data Set

Large	MCAR	< 15%	Univariate	LD, MI or any simple data imputation should work fine but Hot-Deck Methods would be the best choice
			Monotonous	
			Arbitrary	
		> 15% & < 30%	Univariate	LD, or *k*-NN methods would be the best choice
			Monotonous	
			Arbitrary	
		> 30% & < 45%	Univariate	LD, or *k*-NN methods would be the best choice
			Monotonous	
			Arbitrary	
		> 45%	Univariate	Do not impute
			Monotonous	
			Arbitrary	
	MAR	< 15%	Univariate	Combination Method or Methodology
			Monotonous	Combination Method or Methodology
			Arbitrary	Combination Method or Methodology
		> 15% & < 30%	Univariate	Methodology or FIML
			Monotonous	Methodology or FIML
			Arbitrary	Methodology or FIML
		> 30% & < 45%	Univariate	Methodology or FIML
			Monotonous	Methodology or FIML
			Arbitrary	Methodology or FIML
		> 45%	Univariate	Do not impute
			Monotonous	
			Arbitrary	
	NI	< 15%	Univariate	Methodology or FIML
			Monotonous	Methodology or FIML
			Arbitrary	Methodology or FIML
		> 15% & < 30%	Univariate	Methodology or FIML
			Monotonous	Methodology or FIML
			Arbitrary	Methodology or FIML
		> 30% & < 45%	Univariate	Methodology or FIML
			Monotonous	Methodology or FIML
			Arbitrary	Methodology or FIML
		> 45%	Univariate	Do not impute
			Monotonous	
			Arbitrary	

CHAPTER 9 CONCLUSIONS

In this study, our aim was to investigate if prediction accuracies improved and biases decreased when completeness of a data set is enhanced using imputation techniques. We implemented four different imputation strategies (LD, MI, 10 variants of HD, and FIML) on enhancing completeness in incomplete software project data sets. We studied the effects of characteristics of the data set such as size, percentage of data missing, missing data pattern, and missing mechanisms would have on the choice of imputation. We provided a generic classification schema for all software project data sets based on their characteristics. We evaluated their impact by implementing them on six different real-time software project data sets, which are classified into different categories based on their inherent properties. The reliability of the constructed data sets using these techniques was further tested by building prediction models using stepwise regression. We determine the importance of the utilization of these methods by comparison and testing, and find an efficient method for the given class of data set having of missing data.

The most common approach, LD was used in order to compare if other imputation methods performed better. We used MI to test if simple imputation techniques gave better prediction accuracies. We used HD variants because of their broad usage and proven performance. Finally, we used FIML in order to investigate the robustness of model based approaches under different conditions. The results showed that we found a reasonable improvement in the prediction accuracies. The results of our experiments have shown that there was significant improvement in accuracy as well as fitting.

Furthermore, we implemented a hybrid methodology to overcome the limitations in most imputation methods. The methodology initially runs a clustering algorithm on

each of the data sets and forms homogenous clusters. Next, it selects the appropriate cluster as well as the donor(s) from the cluster for each missing case and thus imputes data. To the best of our knowledge we know that no other software engineering researcher has applied clustering techniques to missing value analysis in software data sets. We implemented the methodology on the six different real-time software project data sets and evaluated its performance. We made a comparison study between the proposed methodology and the other traditional methods implemented in this thesis. We also highlighted the conditions to be considered and measures to be taken while using an imputation technique. We noted the ideal and worst conditions for each method and finally detailed on the appropriateness of each method for data imputation.

Our experimental results showed that we succeeded in decreasing bias. The performance of the proposed methodology was the best among all of them. The HD variants and FIML were the best among the traditional approaches implemented. We suggest researchers not to use LD when the data are not MCAR and when missing values are present in a major number of cases but we recommend using MI only when none of the variables singly contribute to a major number of missing values. Also caution should be taken when using MI if the data are not missing at random. On the other hand, HD variants performed well in our analysis. We recommend using variants of HD under MAR assumption. We also suggest using FIML under NI conditions but more thorough testing is needed to confirm its performance.

The proposed methodology was designed by taking into aspect the missing mechanism, data set size, missing percentage and the pattern in which the data are missing. The idea was to overcome the limitations faced by traditional methods. To begin

with, the methodology is not computationally demanding and can work extremely well with all sizes of data sets. Due to the very nature of the method to form homogenous clusters, the missing pattern or the missing mechanism cause no degradation in its performance. However, the only factor that can influence its performance is the percentage of missing data. The performance of the methodology degrades when high percentage of data is missing.

Based on our results, we are sure that we have made a point about the validity of the inferences drawn using traditional approaches. There are only a few references in the literature related to such exploration. Most of them suggest techniques that preserve the integrity of a data set by using different statistical approaches to fill in probable values. In this dissertation, we have contributed towards imputing software project data sets in a new way by using Clustering Algorithms.

9.1 Scope and Future Research

The proposed methodology was tested on six data sets. Though the results drawn from the six data sets gave us confidence on the conclusions, the methodology needs to be tested extensively on a larger number of data sets having high missing percentages (> 40%). Only four kinds of techniques (LD, MI, HD variants and FIML) were used in the study. So other traditional methods can be tried.

Furthermore we encourage implementing other techniques like Bayesian Approaches, Neural Networks etc. The criterion used in the clustering algorithm and its complexity may be improved. Our results are encouraging and we recommend researchers to carry further research using partitional/divisive clustering algorithms, fuzzy clustering on larger number of data sets.

REFERENCES

1) K. Strike, K.E. Emam, and N. Madhavji, Software Cost Estimation with Incomplete Data, ERB-1071 NRC, http:// wwwsel.iit.nrc.ca/~elemam/documents/1071.pdf, also to appear in, IEEE Trans. Software Eng.

2) Roth, P. (1994). Missing data: A conceptual review for applied psychologists. Personnel Psychology, 47, 537-560.

3) Little, R. J. A. and D. B. Rubin, Statistical Analysis with Missing Data. New York, John Wiley & Sons, 2002.

4) M. Berry and M.F. Vanderbroek, A Targeted Assessment of the Software Measurement Process, Proc. IEEE Seventh Int'l Software Metrics Symp., pp. 222-235, 2001.

5) B.W. Boehm, Software Engineering Economics. Prentice Hall, 1981.

6) T. DeMarco, Controlling Software Projects: Management, Measurement, and Estimates, New York: Prentice-Hall, 1982.

7) Myrtveit, I., E. Stensrud and U.H. Olsson, Analyzing data sets with missing data: An empirical evaluation of imputation methods and likelihood-based methods, IEEE Transactions on Software Engineering 27(11), pp999-1013, 2001.

8) Cartwright, M. and M. J. Shepperd, Predicting with Sparse Data, IEEE Transactions on Software Engineering 27(11), pp1014-1022, 2001.

9) R. Park, W. Goethert, and W. Florac, Goal-Driven Software Measurement—A Guidebook, tech. report CMU/SEI-96-HB-002, Software Eng. Inst., Carnegie Mellon Univ., Pittsburgh, 1996.
(www.sei.cmu.edu/publications/documents/96.reports/96.hb.002.html)

10) J. Kim and J. Curry, The Treatment of Missing Data in Multivariate Analysis, Social Methods & Research, vol. 6, pp. 215-240, 1977.

11) K.E. Emam and A. Birk, "Validating the ISO/IEC 15504 Measure of Software Requirements Analysis Process Capability", IEEE Trans. Software Eng., vol. 26, no. 6, pp. 541-566, June 2000.

12) Kalton, G. (1983). Compensating for missing survey data, Michigan, Institute for Social Research.

13) Kalton, G. and Kish, L. (1984). Some efficient random imputation methods, Communications in Statistics, Part A, Theory and Methods, 13, 1919-1939.

14) Schafer, J. L.,&Graham, J.W. (2002). Missing data: Our view of the state of the art. Psychological Methods, 7(2), 147-177.

15) Rubin, D. B. (1976). Inference and missing data. Biometrika, 63, 581-592.

16) B. Cox and R. Folsom, An Empirical Investigation of Alternate Item Nonresponse Adjustments, Proc. Section on Survey Research Methods, pp. 219-223, 1978.

17) L. Ernst, Weighting to Adjust For Partial Nonresponse, Proc. Section on Survey Research Methods, pp. 468-472, 1978.

18) J. Kaiser, he Effectiveness of Hot-Deck Procedures in Small Samples, Proc. Ann. Meeting of the Am. Statistical Assoc., 1983.

19) C.H. Browne, Asymptotic Comparison of Missing Data Procedures for Estimating Factor Loadings, Psychometrika, vol. 48, no. 2, pp. 269-291, 1983.

20) M. Raymond and D. Roberts, A Comparison of Methods for Treating Incomplete Data in Selection Research, Ed. & Psych. Measurement, vol. 47, pp. 13-26, 1987.

21) B. Ford, Missing Data Procedures: A Comparative Study, Proc. Social Statistics Section, pp. 324-329, 1976.

22) S.Y. Lee and Y.-M. Chiu, Analysis of Multivariate Polychoric Correlation Models with Incomplete Data, British J. Math. & Stat. Psych, vol. 43, pp. 145-154, 1990.

23) Kromrey, J., and Hines, C.: Nonrandomly Missing Data in Multiple Regression: An Empirical Comparison of Common Missing-Data Treatments. Educational and Psychological Measurement, vo.54, no.3, pp.573-593 (1994).

24) R.L. Brown, Efficacy of the Indirect Approach for Estimating Structural Equation Models With Missing Data: A Comparison of Five Methods,º Structural Equation Modeling, vol. 1, no. 4, pp. 287- 316, 1994.

25) Roth, P. L., & Switzer, F. S. (1995). A Monte Carlo analysis of missing data techniques in a HRM setting. Journal of Management, 21(5), 1003-1023.

26) Mundform, D. J.,&Whitcomb, A. (1998). Imputing missing values: The effect on the accuracy of classification. *Multiple Linear Regression Viewpoints*, *25*, 13-19.

27) Troxel, A. B., Lipsitz, S. R., & Brennan, T. A. (1997). Weighted estimating equations with nonignorably missing response data. *Biometrics*, *53*, 857-869.

28) Gelman, A., King, G., & Liu, C. (1998). Not asked and not answered: Multiple imputation for multiple surveys. *Journal of the American Statistical Association*, *93*(443), 846-874.

29) Collins, L. M., Schafer, J. L.,&Kam, C. (2001).A comparison of inclusive and restrictive strategies in modern missing data procedures. *Psychological Methods*, *6*(4), 330-351.

30) Sebastiani, P.,&Ramoni, M. (2000). Bayesian inference with missing data using bound and collapse. *Journal of Computational and Graphical Statistics*, *9*(4), 779-800.

31) M.S. Gold and P.M. Bentler, Treatment of Missing Data: A Monte Carlo Comparison of RBHDI, Iterative Stochastic Regression Imputation, and Expectation-Maximation, Structural Equation Modeling, vol. 7, no. 3, pp. 319-355, 2000.

32) Kirsopp, C. and Shepperd, M. *Making Inferences with Small Numbers of Training Sets*, March 2002, Technical Report, Empirical Software Engineering Research Group, Bournemouth University, United Kingdom.

33) Song, Q and Shepperd, M. *A Short Note on Using Multiple Imputation Techniques for Very Small Data Sets*, April 2003, Technical Report, Empirical Software Engineering Research Group, Bournemouth University, United Kingdom.

34) Naoki Ohsugi, Masateru Tsunoda, Akito Monden, Ken-ichi Matsumoto, "Effort Estimation Based on Collaborative Filtering", In *the 5th International Conference on Product Focused Software Process Improvement (PROFES2004)*, Kansai Science City, Japan, pp.274-286, Springer, Berlin Heidelberg, April, 2004.

35) Qinbao Song, Martin Shepperd, Michelle Cartwright and Bheki Twala, *A New Imputation Method for Small Software Project Data Sets* May 2004, Technical Report, Empirical Software Engineering Research Group, Bournemouth University, United Kingdom.

36) J.L. Schafer, Analysis of Incomplete Multivariate Data, Boca Raton: Chapman and Hall, 1997.

37) Martin Shepperd and Michelle Cartwright, Dealing with Missing Software Project Data, November 2002, Technical Report, Empirical Software Engineering Research Group, Bournemouth University, United Kingdom.

38) M. Colledge, J. Johnson, R. Pare, and I. Sande, Large Scale Imputation of Survey Data, Proc. Section on Survey Research Methods, pp. 431-436, 1978.

39) T.W. Anderson, Maximum Likelihood Estimates for Multivariate Normal Distributions when Some Observations are Missing, J. Am. Statistical Assoc., vol. 52, pp. 200-203, 1957.

40) J. Anderson and D.W. Gerbing, The Effects of Sampling Error on Convergence, Improper Solutions, and Goodness-of-Fit Indices for Maximum Likelihood Confirmatory Factor Analysis, Psychometrika, vol. 49, pp. 155-173, 1984.

41) M.C. Neal, Mx: Statistical Modeling, second ed., 1994.

42) Enders, C. K. (2001), A primer on maximum likelihood algorithms available for use with missing data, *Structural Equation Modeling*, *8*, 128-141.

43) Paul D. Allison. Missing Data, Quantitative Applications in the Social Sciences, SAGE Publications, Vol 136, 2002.

44) Arbuckle, J. L. (1996), Full information estimation in the presence of incomplete data, In G. A. Marcoulides & R. E. Schumacker (Eds.), Advanced structural equation modeling (pp. 243- 277). Mahwah, NJ: Lawrence Erlbaum.

45) B. Ford, An Overview of Hot-Deck Procedures, Incomplete Data in Sample Surveys, Theory and Bibliographies, volume 2. W. Madow, I. Olkin, and D. Rubin eds., Academic Press, 1983.

46) I. Sande, Hot-Deck Imputation Procedures, Proc. Symp. Incomplete Data in Sample Surveys, W. Madow and I. Olkin eds., vol. 3, 1983.

47) J. Bailar III and B. Bailar, Comparison of Two Procedures for Imputing Missing Survey Values, Proc. Section on Survey Research Methods, pp. 462-467, 1978.

48) A. Gray and D. MacDonnell, A Comparison of Techniques for Developing Predictive Models of Software Metrics, Information and S/W Tech, vol. 39, pp. 425-437, 1997.

49) D. Heitjan, Annotation: What Can Be Done about Missing Data? Approaches to Imputation, Am. J. Public Health, pp. 548-550, 1997.

50) M.J. Rovine and M. Delaney, Missing Data Estimation in Developmental Research, Statistical Methods in Longitudinal Research: Principles and Structuring Change, A. Von Eye ed., vol. 1, pp. 35-79, New York: Academic, 1990.

51) G. Bohrnstedt and T. Carter, Robustness in Regression Analysis, Sociological Methodology, chapter 5, H. Costner, ed., 1971.

52) Y. Haitovsky, Missing Data in Regression Analysis, J. Royal Statistical Soc., vol. B30, pp. 67-81, 1968.

53) I. Myrtveit and E. Stensrud, A Controlled Experiment to Assess the Benefits of Estimating with Analogy and Regression Models, IEEE Trans. Software Eng., vol. 25, no. 4, pp. 510-525, Jul/Aug. 1999.

54) W. Hayes, Statistics, fifth ed. Hartcourt Brace College Publishers, 1994.

55) T. Mukhopadhyay and S. Kekre, Software Effort Models for Early Estimation of Process Control Applications, IEEE Trans. Software Eng., vol. 18, no. 10, 1992.

56) I. Myrtveit, E. Stensrud, and U. Olsson, Assessing the Benefits of Imputing ERP Projects with Missing Data, Proc. METRICS 2001, pp. 78-84, 2001.

57) E. Stensrud and I. Myrtveit, Human Performance Estimating with Analogy and Regression Models: An Empirical Validation, Proc. METRICS'98, pp. 205-213, 1998.

58) M.J. Shepperd, C. Schofield, and B.A. Kitchenham, Effort Estimation Using Analogy, Proc. 18th Int'l Conf. Software Eng., 1996.

59) A. Myrvold, Data Analysis for Software Metrics, J. Systems and Software, vol. 12, pp. 271-275, 1990.

60) Chipman, H., and Tibshirani, R. "Hybrid Hierarchical Clustering with Applications to Microarray Data", Biostatistics, 2003.

61) Ying Zhao and George Karypis, “Evaluation of hierarchical clustering algorithms for document datasets”, Proceedings of the eleventh international conference on Information and knowledge management (CIKM), Pg 515-524, Mclean, VA, Nov 4-9, 2002.

62) R. Little, Regression with Missing X's: A Review, J. Am. Statistical Assoc., vol. 87, no. 420, pp. 1227-1237, 1992.

63) D.C. Hoaglin, F. Mosteller, and J.W. Tukey, Understanding Robust and Exploratory Data Analysis. John Wiley, 1983.

64) G.A. Milliken and D.A. Johnson, Analysis of Messy Data, Volume 1: Designed Experiments. London: Chapman & Hall, 1992.

65) L.M. Pickard, B.A. Kitchenham, and P. Jones, Combining Empirical Results in Software Engineering, Information and Software Technology, vol. 40, no. 14, pp. 811-821, 1998.

66) W.F. Tichy, Should Computer Scientists Experiment More?, Computer, vol. 31, no. 5, pp. 32-40, May 1998.

VITA

Sumanth Yenduri was born on December 15, 1976, in Rajahmundry, India. He had his preliminary education at Bangalore, India, at Sri Aurobindo Memorial School. He later finished his secondary education and intermediate in Rajahmundry, India, at Sri Siddhartha Residential School and Government Junior College respectively. He further received his Bachelor of Technology in Computer Science and Systems Engineering from Andhra University in 1999. He joined the Department of Computer Science at Louisiana State University as a master's student and received his master's degree in December, 2002. He continued to pursue doctoral studies at LSU. He is expected to graduate with the degree of Doctor of Philosophy during the summer of 2005. His research interests include software process development, data imputation methods, software metrics, software tools, effort/cost/time estimation models, software engineering data analysis and programming languages.