A Framework for Filling of Missing Data: an Imputation Method for Mining and With an Empirical Analysis

K.Fayaz

Abstract: ***Missing data causes a variety of problems in data analysis. First, lost data decrease statistical power. Statistical power refers to the ability of an analytic technique to detect a significant effect in a data set. Second, missing data produce biases in parameter estimates and can make the analysis harder to conduct and the results harder to present. To overcome the missing data problem, a Component-based Framework for imputation is proposed, which efficiently searches the most plausible value for replacement [2]. The algorithm developed ensures complete imputation, the first phase of imputation is based on the complexity of 1) finding the missing value entry, 2) select the category of attribute set, 3) generate the characteristic weight, 4) search the characteristic weight, 5) find the location of missing value and 6) replace the plausible value with missing value.***

Keywords: Missing Values, Imputation, Data Mining, Weight Functions, Component Framework

I. INTRODUCTION

Mathematical Analysis of every module or function is very difficult, instead Empirical Analysis is easy. Empirical Analysis of an algorithm derives the practical time complexity. Asymptotic notations describe theoretical complexity of the algorithm, where by performing empirical analysis, one can easily estimate the amortized complexity of the performance of the algorithm.

The International Software Benchmarking Standards Group (ISBSG) database is no exception. It has a large fraction of missing data, in some variables more than 40 percent. There are several reasons why observations may have missing values. High data collection cost may cause missing values. The cost of gathering and reporting data from software projects is non-negligible. For example, it is more difficult, and, therefore, costly to collect data on Interfaces and Effort than on Users, Sites and Modules. Therefore, we expected,

and did indeed find, that there are more missing values in Interfaces and Effort than in Users, Sites, and Modules. Another reason for missing values is that some values are so called wild values. A wild value is a value we know is untrue.

Revised Manuscript Received on March 10,2019

Dr.K.Fayaz, Department of Computer Science & Technology, S.K. University, Ananthapuram, A.P. India – 515 003

II. RELATED WORK

Data may be missing in any type of study due to any reason. For example, subjects in longitudinal studies often drop out before the study is complete. Sometimes this happens because they are not interested anymore, are not able to find time to participate, died or moved out of the area. Whatever the reason is, the study will suffer from missing-data problem.

Missing data causes a variety of problems in data analysis. First, lost data decrease statistical power. Statistical power refers to the ability of an analytic technique to detect a significant effect in a data set. Also, it is well known that a high level of power often requires a large sample. Thus, it appears that missing data may meaningfully diminish sample size and power.

Second, missing data produce biases in parameter estimates and can make the analysis harder to conduct and the results harder to present. The bias may be either upward or downward, which means the true score may be either overestimated or underestimated.

Imputation: "The action of attributing something, usually a fault or a crime to someone."

Hitherto, there are methods that deal with missingness, they are mean imputation and multiple imputations. Imputation consists of replacing the missing data with values derived from the respondents or from a relationship between the nonrespondents and respondents.

According to Little and Rubin (2002), the mechanisms leading to missing data can be classified into three subgroups:

- Missing completely at random (MCAR)
- Missing at random (MAR) and
- Not missing at random (NMAR).

MCAR means that the missing data mechanism is unrelated to the variables under study, whether missing or observed: a missing response happens purely by chance.

That is f (M | Y,Q) □ □ f (M | Q) for all Y,Q .

Let Yobs denote the observed components of Y and let Ymis denote the missing components[4].

The Framework

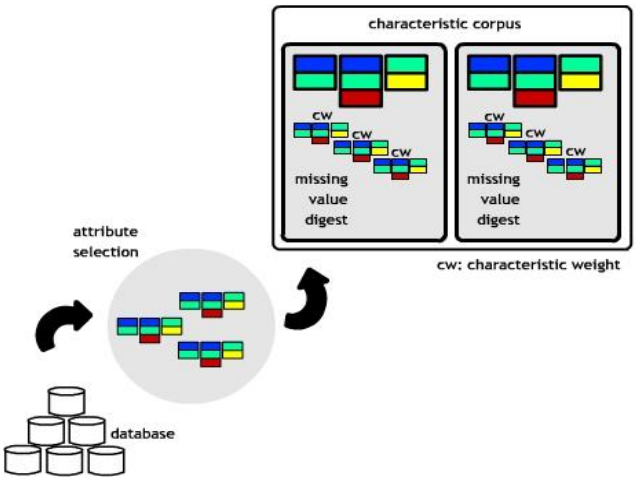

Figure: Imputation Framework using Characteristic Corpus and MVD

III. Information-hash theoretic approach:

Generally Information Theoretic approach follows the information about the attributes that describe their characteristics with more efficacies that is required for solving the problem. Information-hash theoretic approach, deals with the collection and various combination of the attributes that contain maximum frequency of missing values. An attribute in the tuple that is earmarked as having missing value in other tuples is said to be called existent attribute of that tuple. An attribute in the tuple is having missing value is called a missing attribute of that tuple. The combination of these attributes can be set according to the domain interest or the user choice which eases the algorithm complexity. More number of possible combinations of existent attributes is built for making out more number of possibilities to guess the missing value in the tuple. The hashing mechanism adopted in this algorithmic approach proposed generating a quantum figure for describing the characteristics of a tuple with respect to the missing value attributes and existent attributes.

AXIOM 1: Selection of the attributes into the attribute set is based on the frequency of missing values. The attribute containing less number of missing values is selected for the attribute set.

IV. Proposed Algorithms

4.1 Generate the Characteristic Corpus:

The characteristic corpus is contained with attribute, value. This is a key mechanism used to replace a particular missing value in the original tuple of the sampled database. This is also used to calculate the **Missing Value Digest.** The Characteristic Weight Corpus is filtered with the attribute and MVD is found. The corresponding value is considered as the missing value and is imputed in the original database.

4.2 Generating MVD:

MVD is generated with various alternatives of attributes in the database. Alternatives of attributes include, single attributes, combination of attributes. Various MVDs are generated for the collection of existent attributes set from the sampled database. MVDs bring out various possibilities that a tuple can be found with various combination of attributes that contain missing values. All the possible MVDs are stored in the characteristic corpus. For the tuple in the database that is generated:

1) Selection of attribute set
2) Frequency count of attribute set
3) Calculate the characteristic weight for each attribute set based on frequency count
4) MVD is calculated based on rank of each characteristic weight of attribute set.
5) **MVD association** is developed.

4.3 Experimental Evaluation of Algorithms

In this work a Java-based Efficient Imputation System that can be easily deployed on any Java virtual machine (JVM) platform and gathered datasets from several online data repositories as the data source. The experiments tested the viability of the three major components in this work, namely, our new term weighting scheme and our new sentence-modeling scheme.

Throughout, each successive design proposal is evaluated after describing that proposal in order to isolate its performance impact and motivate further refinements. The total work is practically carried out using Java SDK on mushroom dataset. The previous works of the imputation algorithms are experimented and tested in C/C++ on Linux, their performance is compared with proposed experiment.

4.4 Data Selection:

To meet the experimental feasibility of the work, the imputation procedure has been carried out as follows: The database consists of entities with tuples, where some of the tuples contain the missing attribute values. The tuple to be imputed by the tool developed in this work is given as a query tuple and placed in a separate file. The whole data (original data) that contains the imputed tuple with all other tuples is given as training data set to the tool. The query tuple is stored in `query.txt` and whole original data as `dataset.txt`. The mushroom data set consisting of 23 attributes, 8124 tuples inclusive of missing attributes is pre-processed by the tool for about 10 minutes and processed for imputation in 0 to 1 seconds.

Data set

The data set for this assignment consists of descriptions of mushrooms drawn from the Audubon Society Field Guide. Each mushroom is described by 22 physical characteristics (e.g., scaly, yellow) and also by whether it is poisonous or edible.

Information about the data base can be obtained at `ftp://ftp.ics.uci.edu/pub/machine-learning-databases/mushroom`

The file agaricus-lepoita.names contains information about the data base and the 22 attributes, and the file

agaricus-lepiota.data contains the actual data. The data set contains some records with missing values (indicated by question marks). These items are removed from the data set, and split the data into a training set and a test set. You should retrieve the data from the site:

ftp://ftp.cs.colorado.edu/users/mozer/5622/mushroom.train
ftp://ftp.cs.colorado.edu/users/mozer/5622/mushroom.test

The first column of the data contains the class label, "p" for poisonous and "e" for edible.
Various sites have been visited to collect the bench mark data to test the propensity of the tool. Some of the links are given below:

(a) http://www.nd.edu/~oss/Data/data.html
(b) http://kdd.ics.uci.edu/summary.data.type.html
(c) http://data.eol.ucar.edu/codiac/ds_proj?Operational
(d) http://www.lshtm.ac.uk/msu/missingdata/biblio.html

- **. Application Development**

A swing based application is developed to implement complete imputation model of the work. Dataset are read into the application and represented by the collection utilities of JDK. This is to avoid the time consumption during disk input output and to amortize the overhead of time complexity in overall algorithmic implementation. The application consumes considerable amount of time in preprocessing. The preprocessing stage includes, converting the file data into the collection utility framework, selection of attributes relevant for the imputation, and various samples of attributes for checking the highest probability of the value that is to be imputed. While converting the file data into the collection utility framework of the platform, the digest values are generated and stored in the repository. The tool provides the walkthrough into the corpus available. The missing value digest, which is a collection of selected characteristic weights of the tuple are represented as corpus. This helps in the algorithm for searching the more relevant tuple that meets the needs of finding the plausible value for imputation. The tool EfficientImputer is developed in the present process of ongoing research work. The total experiment is carried our in three stages.

- **. Attribute Selection Analysis and Imputation**

For this analysis 3 experiments have been conducted. All the experiment listed below explains the various tasks of the imputation tool illustratively. First all the list of attributes are considered for generation of digest values and then allowed for selection. Subsequently, the numbers of attributes that are going to participate actively are not chosen from the user input. Without the numbers of attributes the scope of generating the digest values and storing the corpus becomes huge and the search space for the prime imputation mechanism becomes vast leading to insignificant loss of time. If the numbers of attributes that are going to participate actively are chosen from the user, then the search space consideration is limited to a feasible extent. Various samples of the attributes are selection as a part of attribute selection, stating the possibility idea to obtain the plausible value for imputation. Even though the attribute selection is manually taken care, it is adjudged by the relevant knowledge of the experimenter about the tuple and the domain of the selected dataset.

Experiment 1:

Base Test Case:

Original Data	8124 tuples (23 species of gilled mushrooms in the Agaricus and Lepiota Family)
Query Data	e,x,y,u,f,n,f,c,n,w,e,**?**,s,f,w,w,p,w,o,f,h,y,d
Scope Attributes Considered	12
Selections	3 (Three Attribute Selections) (1) 9, 10, 12 (2) 10, 11, 12 (3) 1, 5, 10, 12
Results:	Position of **?** is 11th in the query tuple Attribute Selection number **(2)** is ignored, since it contains (11th) attribute reference to the **?** in the query tuple. Plausible value is searched using **(1)** and **(3)** attribute selections. Tuple **#12** contains plausible value. Value imputed is '**c**' at '**?**'

The tool developed is illustrated briefly with all screens and

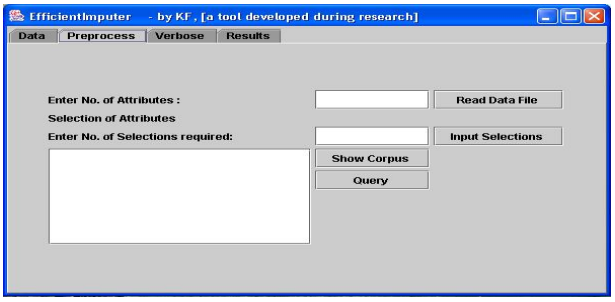

Figure: A Typical Screen containing inputs of all parameters of `EfficientImputer`

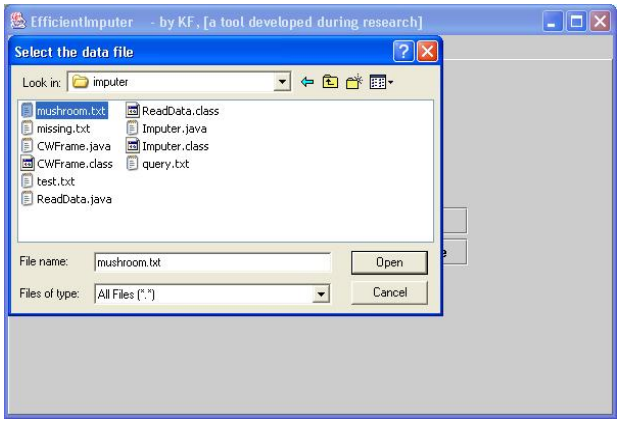

Figure: `EfficientImputer`: Loading and Reading Datasets and Query

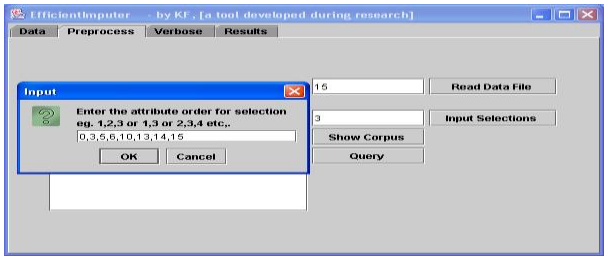

Figure: `EfficientImputer`: Attribute Selection

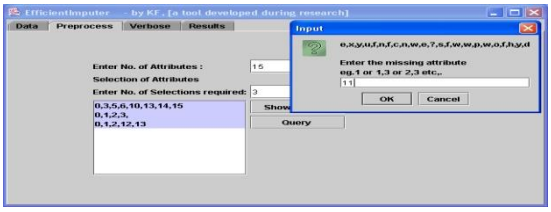

Figure: `EfficientImputer`: Triggering Query omitting the attribute that contains the missing value i.e., ?.

The corpus is generated for all the data in the experiment. The Characteristic Weights and Missing Value Digest of the tuples participate in generating the corpus required for the experiment. The corpus is the basic search space to find the plausible value for the imputation. Characteristic Weights are described using Digest Values (a variant of hash values), bitmapped equivalent values of the data set. Several items in the data set are identified and the radix of the data set is calculated. Based on radix an **n**-bit string is assumed to posses the binary values indicating the each item of the dataset uniquely. **n** is calculated as follows:

Let r be radix

$$\mathbf{n} = \sqrt{r}$$

If r is not a perfect square, then **n** is incremented by 1 of its integer part and n is assumed as int(**n**)+1. Where int(**n**)+1 bits are used to represent the bit string to assume the digest value of the item in the dataset.

In the experiment the data set used contains 23 attributes including the p or e attribute which is the first attribute of the tuples.

No. of items = 23 i.e., r = 23

$$\mathbf{n} = \sqrt{r}$$

$$\mathbf{n} = 4.795831523312719541597438064 1627$$

No. of bits in the bit-string is int(n) + 1 i.e., 4 + 1 = 5

5 bits are used to form a bit-string that can accommodate the digest value of all the items in the current data set.

Following is the verbose output of the experiment conducted

0,1,2
0,3,4,5
2,3,4
Selected Indexes 3
0**
1**
2**
LIST OF SELECTIONS
3

0,1,2
Query digest to locate in [CW] MVD Corpus
00010:00001:00011:
* * * * *
00001:00010:00011:
00010:00011:00001:
00001:00010:00101:
00010:00001:00011:
Located at Record #3

Corpus for Attribute Selections shown in the Verbose output of the experiment

No. of attributes considered for experiment : 6
No. of Attribute Selections: 3
Attribute Selection-1: 0,1,2
Attribute Selection-2: 0,3,4,5
Attribute Selection-3: 2,3,4

Selection : 1
00001:00010:00011:
00010:00011:00001:
00001:00010:00101:
00010:00001:00011:
00011:00010:00001:
00110:00011:00001:
***

Selection : 2
00001:00100:00101:00110:
00010:00011:00100:00001:
00001:00100:00011:00110:
00010:00011:00100:00001:
00011:00100:00101:00110:
00110:00011:00101:00100:
***
Selection : 3
00011:00100:00101:
00001:00011:00100:
00101:00100:00011:
00011:00011:00100:
00001:00100:00101:
00001:00011:00101:
***

In the above verbose output, the selection-1, selection-2, selection-3 are the attribute selection sequences that are used to find the plausible value for imputation. As it is very tedious to locate the plausible value in the entire database, according to users' certain guesses or based on the domain knowledge (background information) of the user, the attribute selection sequences can be generated. This is a sampling activity where the user has to be aware of the attributes and their positions which are having higher probability of having the plausible values for the imputation. Such attributes are selected and combined with various other attributes as various types of attribute selections, for example in the above verbose output, the three attribute selection sequences are 0,1,2; 0,3,4,5; 2,3,4. The 0,1,2 indicates the first attribute selection that contains the combination of attributes 0, 1, 2. (The position of the first attribute is identified as to be at the position 0). Similarly, 0,3,4,5 means the attribute selection later considered, that contains the combination of attributes 0,3,4,5. This illustrates that the attribute considered for the attribute selection not necessarily be in the sequence, they may be selected in assorted style also, except that all the item positions are organized in the order. 0, 3, 4, 5 or 2, 3, 4 are valid attribute selections, but 0,3,2,4 is not a valid selection. 0,3,2,4 is also an assorted selection of attributes but not organized in the ascending order.

The search and replace procedure of the missing value with plausible value is carried out with specially developed API based code. Many fragments of code are written to represent the logical idea of the problem and it's solving procedure. As seen in the previous example experiment, the data set is converted into digest data set, where all the digest values of the items in the data set are present. Digest values are not necessarily compressed bitmap representation; rather, these values especially exhibit the characteristic features of the items in the data set. Digest values are generated for all the items of the data set and further they are organized in various sequences of bit strings based on the attribute selection sequences.

The attribute selection sequence governs the generation of the characteristic weights of items in each tuple, subsequently combining all the characteristic weights of the items i.e., digest of the tuple based on the attribute selection, are collected as missing value digest.

The searching part has been coded diligently to enable the search choose sequential approach for small data sets and faster search with pattern matching in the case of huge data sets, thus by regulating the time complexity.

```
String getBits(String s)
{
 String bits;
 Character cs = new
Character(s.charAt(0));
 switch(cs.charValue())
 {
   case 'a': bits="00001"; break;
   case 'b': bits="00010"; break;
   case 'c': bits="00011"; break;
   case 'd': bits="00100"; break;
   case 'e': bits="00101"; break;
   case 'f': bits="00110"; break;
   case 'g': bits="00111"; break;
   case 'h': bits="01000"; break;
   case 'i': bits="01001"; break;
   case 'j': bits="01010"; break;
   case 'k': bits="01011"; break;
   case 'l': bits="01100"; break;
   case 'm': bits="01101"; break;
   case 'n': bits="01110"; break;
   case 'o': bits="01111"; break;
   case 'p': bits="10000"; break;
   case 'q': bits="10001"; break;
   case 'r': bits="10010"; break;
   case 's': bits="10011"; break;
   case 't': bits="10100"; break;
   case 'u': bits="10101"; break;
   case 'v': bits="10110"; break;
   case 'w': bits="10111"; break;
   case 'x': bits="11000"; break;
   case 'y': bits="11001"; break;
   case 'z': bits="11010"; break;
   default:  bits="00000";
```

```
  }
  return bits;
 }
```

The above code fragment is supported by **getDigest("a")**

```
 String getDigest(String line)
 {
  int i=0;
  String d="";
  String r[] = line.split(",");
  for(i=0;i<r.length;i++)
    d+=getBits(r[i]);
  return d;
 }
```

The generated digest are stored in the Vector collections and further are used in the process of locating the pattern of query in the data sets. The Characteristic Weights are stored in the Vector collection, which contains the digest values of the attributes that belong to the selected attribute selections only. The Vector that stores Characteristic Weights is an array that contains number of elements equal to the number of attribute selections. These attribute selections are considered in the imputer during the input provided by the user at the time of selecting the query trigger. When the query trigger button is selected, the immediate input to be given is the position of the attribute in the query tuple that contains the missing value i.e., **?**. As soon as the position is input, the attribute selections that contain the position as the subset will be simply ignored for the search space. Only the digest values of those attribute selections that does not contain the position of the missing value in the query tuple are allowed search for the plausible value.

The code fragment **replaceAttribute**, shown below illustrate the procedure of finding the position of the missing value, identifying the plausible value and replacing the missing value with the plausible value.

```
 public String replaceAttribute(String q,
String t)
 {
  String res="";
  int pos = q.indexOf("?");
  char s = t.charAt(pos);
  StringBuffer qs = new StringBuffer(q);
  qs.replace(pos,pos+1,s+"");
  res = qs.toString();
  return res;
 }
```

V.. Type of data in the attributes

Attributes that are selected for the process are alphanumeric in type of one byte width. If the value is more than one byte width, then equivalent bitmapped values are generated for the data set in the attribute. The bitmapped values can be assumed for any type of attributes. For example if the data set in the attribute are {positive, negative}, then the equivalent bits for the values can be taken and digest value can be generated.

{Positive, negative} = {0, 1}

Data set related to Hepatitis Domain is another example, where the attribute class is considered for processing. As the number of values in the attribute class is only two a binary digit can be considered for generating a digest value.

```
1. Class: DIE, LIVE
2. AGE: 10, 20, 30, 40, 50, 60, 70, 80
3. SEX: male, female
4. STEROID: no, yes
5. ANTIVIRALS: no, yes
6. FATIGUE: no, yes
7. MALAISE: no, yes
8. ANOREXIA: no, yes
   .....
   .....

18. ALBUMIN: 2.1, 3.0, 3.8, 4.5, 5.0,
6.0
19. PROTIME: 10, 20, 30, 40, 50, 60, 70,
80, 90
20. HISTOLOGY: no, yes
```

{DIE, LIVE} = {0, 1}

Apart from the conversion of the one-byte alphanumeric data into bits, the string or word data can be converted into bits (digest values) by considering the bitmap. A bit map is designed dynamically and a bitset are constructed to design the digest value. By using the basic structure of the data set i.e., attributes information it will be easy to generate the bit map and the equivalent bitset to generate the digest value of the attribute and further it can be merged with the whole digest value of the tuple.

VI.. Case Wise Analysis

The core parameters of the experiment are number of records, number of attributes, and position of the attribute that contains the missing value in the query tuple. The variable parameters of the experiment that change, to influence the efficiency and time for finding out the plausible value are, number of attributes that are considered for the whole imputation process, number of attribute selection sequence samples, processing time (in msec.).

No. Of Records	No. Of Attributes	Position Of Missing Value	No. Of Attributes Considered For The Whole Imputation Process	No. Of Attribute Selection Samples	Processing Time
8124	16	11	13	4	0.45
8124	16	11	13	3	0.35
8124	16	11	13	2	0.3
8124	15	11	13	4	0.35
8124	15	11	13	3	0.3
8124	15	11	13	2	0.2
8124	14	11	13	3	0.3
8124	13	11	13	2	0.175

Table: Table showing the Parameters for various experiments using `EfficeintImputer`

The performance of the imputer changes according to the core input and the variable inputs provided during the experiment. The core input remain constant as the data set is not changed in the experiment, where the variable inputs change, as the thirst of find the plausible value as quickly as possible increases. The process of finding of the plausible value is narrowed based on the number of observations made in the experiment. However, this requires the knowledge about the domain of the data set or the higher probabilistic rates of guess. As the guess is not the measure to consider the attribute selections, various possibilities of attribute selection sequences are made. By considering the combinations of the items in the tuples of the data set, various attribute selection sequences are constructed.

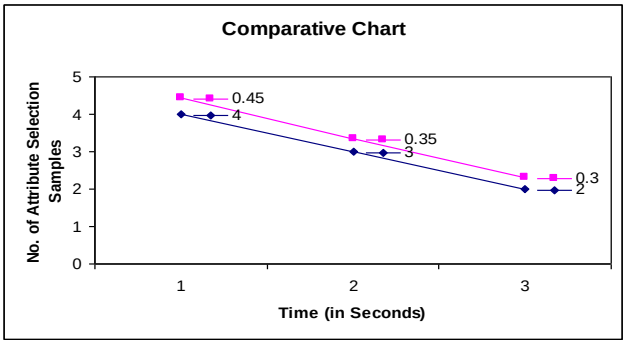

Figure: A Comparative Chart showing No. of Attribute Selection Samples vs. Time [A]

The above comparative chart is illustrating the curve with proportionate increase of time based on the increase of the number of attribute selections. For the above; the number of tuples selected in 8124, Number of attributes that are selected for the experiment is 16, and Position of the missing value in the query tuple is 11th position, Number of attributes that are participating in the current instance of experiment is 13, number of attribute selection samples and the time taken to process the imputation are graphed.

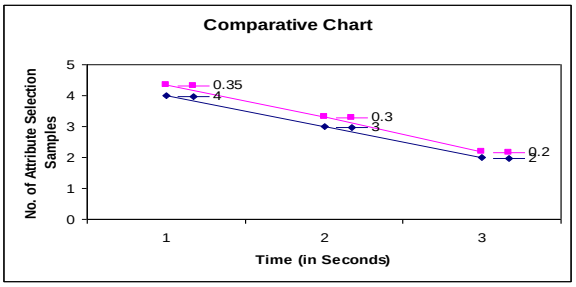

Figure: A Comparative Chart showing No. of Attribute Selection Samples vs. Time [B]

The above comparative chart is illustrating the curve with proportionate increase of time based on the increase of the number of attribute selections. For the above; the number of tuples selected in 8124, Number of attributes that are selected for the experiment is 15, and Position of the missing value in the query tuple is 11th position, Number of attributes that are participating in the current instance of experiment is 13, number of attribute selection samples and the time taken to process the imputation are graphed.

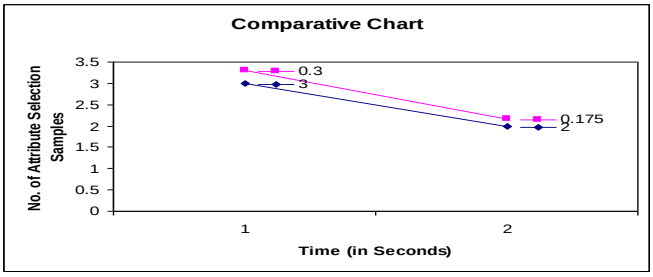

Figure: A Comparative Chart showing No. of Attribute Selection Samples vs. Time [C]

The above comparative chart is illustrating the curve with proportionate increase of time based on the increase of the number of attribute selections. For the above; the number of tuples selected in 8124, Number of attributes that are selected for the experiment is 14 and 13, and the Position of the missing value in the query tuple is 11th position, Number of attributes that are participating in the current instance of experiment is 13, number of attribute selection samples and the time taken to process the imputation are graphed.

VII. Observation from Graphs

All the above graphs show the two parallel or disjoint curves. This indicates, there is a proportional difference between the number of attribute selection samples and the time. From the above observation it may be concluded that "The rate of increase of attribute selection samples influences the increase of time".

VIII. Evaluation

In this section, we report a systematic empirical study using real data sets and synthetic data sets. All the experiments were conducted on a PC computer running the Microsoft Windows XP Professional Edition operating system, with a 3.0 GHz Pentium 4 CPU, 1.0 GB main memory, and a 160 GB hard disk. Our algorithms were implemented in Java. By default, our method was implemented.

The greedy search method with the help of contribution score, and set the order q = 1 in the correlation based sample quality score.

IX. . Sampling

Sampling methods are classified as either probability or nonprobability. In probability samples, each number of the population has a known non-zero probability of being selected. Probability methods include random sampling, systematic sampling, and stratified sampling. In nonprobability sampling, numbers are selected from the population in some nonrandom manner. These include convenience sampling, judgment sampling, quota sampling, and snowball sampling.

An optimistic method of sampling is implemented in order to select the sample tuples that contains all tuples having the missing values and adequate number of tuples that propose the suggestion for the plausible values; however they are algorithmically matched by the characteristic weights.

9.1 Attribute Selection

Selecting the attributes into the attribute sets; the attribute set plays a very important role, in the overall framework. As per the definition given to the attribute set, the set of attributes which are most likely posses no missing value are selected as members of the attribute set, such sets having existent attributes are built. Number of iterations has been performed to understand the attribute as an existent attribute. In the strict case, the attribute even having a single missing value will become a non-existent attribute. The existent-attributes are grouped together into several groups based on the information provided by the general characteristics and background knowledge about the attributes. A simple iterative heuristic algorithm is implemented to perform attribute selection successful.

9.2 Generating Characteristic Weights

Generating Characteristic Weights is process of developing unique digest values using a hashing algorithm. The hash algorithm generates the digest value for all the tuples that does not contain missing values. The digest values are generated for each set of attributes.

9.3 Implementing Search

The record where the missing value is available is identified. In order to start imputation, the possible set of attributes possible from the record is prepared. For all the attribute sets, the characteristic weight (hash value) is generated and the CW is searched from the MVD of that attribute set from the characteristic corpus. The CWs of the attribute set are arranged in a list with sorted order. The CW is searched on the sorted data using a binary search technique and the corresponding record position is located.

X. . Framework Reliability

"Reliability is the consistency of a set of measurements or measuring instrument, often used to describe a test. This can either be whether the measurements of the same instrument give or are likely to give the same measurement (test-retest), or in the case of more subjective instruments ".

The components and their integration are optimistic. The connection between the components is reliable with regard to the transactions between them. The reliability in this research work is deposited in two fold.

First, in maintaining the Characteristic Corpus. The characteristic corpus is a collection of CWs of various categories. For the generation of characteristic weights it is compulsory that to select the set of attributes. If the CW is calculated for the entire set of attributes, the complexity of storing the MVD increases. For a database having 1500 attributes, typically the database belongs to Cancer, DNA structures etc, The CW calculation will be ineffective and not suggested to be reliable. Since, there are chances of duplicate CWs generated for al huge set of attributes, the attributes are grouped, and this is a reliable measure implemented to generate CWs uniquely with various categories of attributes.

Second fold of reliability is, implementing the search and replacement of plausible value in the place of missing value. An efficient search mechanism for sorted data is implemented. The CWs are sorted and the search and replacement algorithm is performed on the CWs list, to locate the suitable record containing the plausible value.

10.1 Experimental Setup

The mining techniques proposed in the current research work are implemented and evaluated. The language used was Java and the experiments were performed on a Pentium IV 3.0 GHz with 1GB of memory, running Windows XP Professional. Some of the important phases of the framework have been implemented using Orange tool, which developed in Python language. Python is a variant of Object Oriented Programming Languages like C++ and Java. Effective GUI tool Orange presents widget-oriented approach for:

1. Sampling the data set
2. Visualizing the sample dataset

The data sample is very essential for testing the framework of imputation proposed in the research work, for various alternatives of samples of data. Unlike in the single and multiple imputation techniques, which propose a stochastic process (random variable set) as the result of experiment to be replaced in the data in the missing places, the framework proposes more exactly fitting value for the missing locations.

Data samples are very effectively presented visualization widgets of Orange tool. Distributions, Attribute Statistics, Scatterplot, Linear Projection, Radial Visualization, Polynomial Visualization, Parallel Coordinates, Survey Plot, Mosaic Display and Sieve Diagram. Radial visualization and Parallel Coordinates has been effectively used to present samples extracted from the dataset.

XI. Generating Corpus

Corpus is an important storage in this framework. All the digest values are stored in the corpus orderly. The data structures are implemented meticulously to draw each digest value carefully for searching and replacing. The weights which are developed from each tuple are represented in a categorical attribute set called as Missing Value digest, a separate data structure (grouped) is implemented to store all CWs as a MVD. All the categorical vectors of MVDs are collectively represented as Corpus.
Hashing algorithms are implemented to generate the weight values for each tuples.

XII. Imputation

This is a two phase algorithm. In the first phase, (searching) the tuple for replacing the missing value is selected, selecting the attribute set (category of attributes) is selected and a CW is generated. The CW is searched in the Corpus, The MVDs which reflect the values of CW are selected and the entry positions of the data set is found, which contains the plausible value to replace in the missing value location. The second phase, the plausible value to be replaced is found and the same is replaced in the missing value location.
The algorithm complexity of the first phase of imputation is based on the complexity of 1) finding the missing value entry, 2) select the category of attribute set, 3) generate the characteristic weight, 4) search the characteristic weight, 5) find the location of missing value and 6) replace the plausible value with missing value.

XIII. . Conclusion and Future Work

It is a great challenging bridging activity of software and technology development and social needs. It is claimed that one of the main obstacles is the underlying “metaphor” of “delivery”, and assumption of social scientists potential of presenting relevant ideas and empirical analyses to meet the expectations that lead to many practically insights.

REFERENCES

1. Andrew Kusiak, Data Mining and Decision Making, Proceedings of the SPIE Conference on Data Mining and Knowledge Discovery: Theory, Tools, and Technology IV, Vol. 4730, April 2002.
2. Emanuele Olivetti and Paolo Avesani, Active Sampling for Data Mining, ITC-iRST: Via Sommarive 14 – I-38050 Povo (TN) - Italy, May 2001.
3. Insightful Corporation, Analyzing Data with Missing Value in S-Plus, Seattle, Washington , September 2001.
4. Ito Wasito, Least Squares Algorithms with Nearest Neighbour Techniques for Imputing Missing Data Values, University of London , April 2003.
5. Jiawei Han, Micheline Kamber, Data Mining: Concepts and Techniques, Elsevier Publications, 2007.
6. Jing Zhou, The Missing Value Problem: A Review And Case Study, University of Maryland, College Park, 2006.
7. Lori Bowen Ayre, Data Mining for Information Professionals , June 2006
8. Margaret H Dunham, S. Sridhar, Data Mining: Introductory and Advanced Topics, Pearson Education, 2006
9. Paul Kofman, Ian Sharpe, Imputation Methods for Incomplete Dependent Variables in Finance, University of Technology, University of New South Wales, Sydney., January 2000.
10. Rayner Alfred, Knowledge Discovery: Enhancing Data mining and Decision Support Integration, University of York , Jul-05
11. Roderick JA Little, Donald B Rubin, Statistical Analysis with Missing Data, Wiley Publications, 2002.
12. Yoshikazu Fujikawa, Efficient Algorithms for Dealing with Missing Values in Knowledge Discovery, Japan Advanced Institute of Science and Technology, February, 2001.
13. Bill Gates, {has quoted} Data Quality, extract from Data Strategies in Companies, Chapter 3 pp. 47.