DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

DATA MINING

SUMMER 2019

GENERALIZED REGRESSION NEURAL NETWORK ENSEMBLE FOR SINGLE IMPUTATION [GESI]

PROFESSOR

PROF. ROOZBEHRAZAVI -FAR

SUBMITTED BY:

Jheel Patel	104924291
Darshpreet Kaur	104978548

Submission Date: 06-July-2019

TABLE OF CONTENTS

TABLE OF FIGURES

ABSTRACT

The project deals with development of MATLAB code for single imputation of missing data using a generalized regression neural network for an unlabelled numerical dataset. The incomplete data set will be divided into a complete and incomplete dataset. Relieff algorithm will be applied to choose the optimal subsets. The subsets shall be used to train the GRNN, and the model will be fed to impute the missing data. NRMS will be calculated by comparing the imputed data sets with the original data sets. Moreover, the algorithm shall be analyzed for runtime against the data size and dimensions.

1. INTRODUCTION

Missing data is common in large data and occurs due to nonresponse of certain respondents due to lack of information, or unwillingness to share. These form of missingness takes different types and different impacts on the analysis.

There are basically three major forms of missing values

1. Missing at Random(MAR)
   Example: One of the gender may be less likely to disclose their weight
   Probability that Y(output) is missing depends only on the value of X(Input)

2. Missing Completely at Random(MCAR)
   Example: There is no particular reason why some respondents disclose their weights and others do not;
   Probability that Y is missing depends only on the value of X

3. Missing Not at Random(MNAR)
   Example: Light (or heavy) persons are less likely to disclose their weight
   The probability that Y is missing depends on the unobserved value of Y itself.

There are two major methods to deal with missing value, listwise deletion and Imputation. Listwise deletion removes all data for the row having one or more missing value. However, this leads to reduction in the size of complete dataset. Hence, its important to impute the missing values.

The imputation can be single or multiple. In single imputation, each missing value is computed in a each for loop, whereas in multiple imputation generate multiple datasets, perform statistical analysis on them.

The method of GESI bifurcates the data into complete data and incomplete dataset. Then complete dataset is then used to train GRNN and the trained values are used to impute the missing dataset. To impute the missing value, we consider the column contain missing value as our output and the other columns are input. However, it becomes computationally time consuming in case of a large no of features. Hence, we apply Relieff algorithm to select 10 optimal feature subsets as input, which reduces computational time.

2. LITERATURE REVIEW

The Journal Article “A neural network-based framework for the reconstruction of incomplete data sets” by Iffat A. Gheyas n, LeslieS. Smith proposes an algorithm Generalised Neural Network Ensemble for Multiple Imputation as well as single imputation of the version-GESI. The author compares the algorithm with 25 popular missing data imputation algorithms. The major focus of the research was to develop an imputation algorithm that preserves the

multivariate joint distribution of the input and output variables. The algorithm is non-parametric. [1]

3. GENERALISED REGRESSION NEURAL NETWORK(GRNN)

GRNN is a neural network-based function approximation or estimation algorithm that predicts the output of a given input data.

The neural network needs training data, which contains input-output mapping to train itself. The network will train itself with the training data set, and when we feed a new testing data set, it will accordingly give the output or predict the result.

In the case of GRNN, the weighted average of the outputs of the training dataset is used to estimate the output. Here the Euclidean distance between the training data and test data is used to calculate the weights. If the weight or distance is small, then the weight will be more on output, and if the distance is large, it will put less weight to the output. [2]

3.1 NETWORK ARCHITECTURE

Network architecture contains four basic layers. (1) Input layer, (2) Pattern layer, (3) Summation layer, and (4) Output layer.

Input layer: It feeds the input to the next layer.

Pattern layer: It calculates the activation function and Euclidean distance.

Summation layer: It has two subparts one is Numerator part and Denominator part. The Denominator is the summation of all activation function whereas the numerator is the summation of the multiplication of training output data and activation function. It feeds both the Numerator & Denominator for the next output layer.

Output layer: It contains one neuron which calculates the output by dividing the numerator and the denominator part. [2]

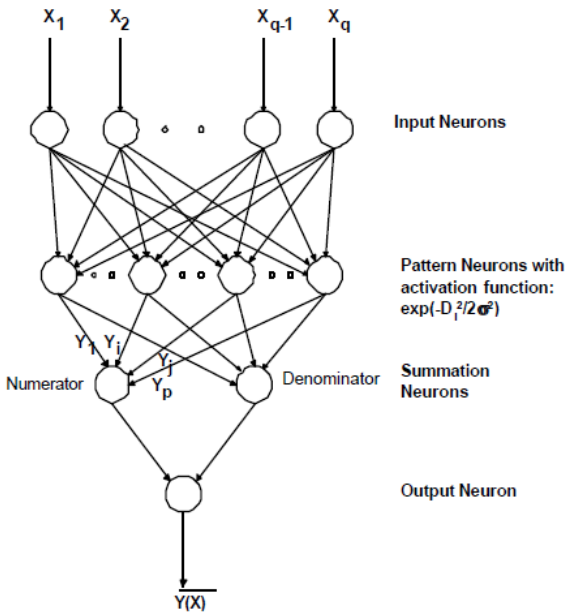


Figure 1 : Network Architecture of GRNN [2]

3.2 WORKING

GRNN stands on the below equation:

$$Y(x) = \frac{\sum_{i=1}^{n} Y_i \exp\left(\frac{-D_i^2}{2\sigma^2}\right)}{\sum_{i=1}^{n} \exp\left(\frac{-D_i^2}{2\sigma^2}\right)}$$

Where, $D_i^2 = (X - X_i)^T . (X - X_i)$

Here $\boldsymbol{X}$ is the input sample and $\boldsymbol{X_i}$ is the training sample. Output of the input sample **i** is $\boldsymbol{Y_i}$ $\boldsymbol{D_i^2}$ is the Euclidean distance from the $\boldsymbol{X}$ and $\boldsymbol{X_i}\mathbf{exp}\left(\frac{-\boldsymbol{D_i^2}}{2\sigma^2}\right)$ is the activation function. Basically, this activation function theoretically is the weight for that input.

The value of $\boldsymbol{D_i^2}$ signifies how much the training sample can contribute to the output of that test particular test sample.

If $\boldsymbol{D_i^2}$ has small value that means it will contribute more value to the output but if it has a big value that means it will contribute very less to the output.

The term $\mathbf{exp}\left(\frac{-\boldsymbol{D_i^2}}{2\sigma^2}\right)$ is deciding that how much weight the training sample will contribute.

If $\boldsymbol{D_i^2}$ is small value, the term $\mathbf{exp}\left(\frac{-\boldsymbol{D_i^2}}{2\sigma^2}\right)$ returns a relatively large value.

If $\boldsymbol{D_i^2}$ is large value, the term $\mathbf{exp}\left(\frac{-\boldsymbol{D_i^2}}{2\sigma^2}\right)$ returns a relatively small value. [2] [3]

3.3 EXAMPLE

Input	Output
2	3
4	5
6	7
8	9

Table 1 : Example of given dataset for GRNN

What will be output for 5?

Step1 : Calculate $D_1 = (5-2)^2 = 9 \mid D_2 = (5-4)^2 = 1 \mid D_3 = (5-6)^2 = 1 \mid D_4 = (5-8)^2 = 9$

Step2 : Calculate weights using the activation function: $\mathbf{exp}\left(\frac{-\boldsymbol{D_i^2}}{2\sigma^2}\right)$

Let $\boldsymbol{\sigma = 1}$

So, weights are, $W_1 = 0.01$ | $W_2 = 0.6$ | $W_3 = 0.6$ | $W_4 = 0.01$

Step 3 : Summations of W's= $W_1 + W_2 + W_3 + W_4 = 1.22$

So, denominator is **1.22**

Now numerator is YW = $W_1 * Y_1 + W_2 * Y_2 + W_3 * Y_3 + W_4 * Y_4$

$= 0.01 * 3 + 0.6 * 5 + 0.6 * 7 + 0.01 * 9 = \mathbf{7.32}$

Step 4 : So, the output is: (Numerator/Denominator)

Output = $\frac{\sum W*Y}{\sum W} = \frac{7.32}{1.22} = 6$

So predicted output is **6.**

3.4 ADVANTAGES OF GRNN

1. The main benefit of GRNN is it speeds up the training process, which helps the network to be trained faster.
2. The network learns from training data by "1-pass" training in a fraction of the time it takes to train standard feedforward networks.
3. GRNN estimation always converges to a global solution instead of being trapped by a local minimum unlike standard feed forward networks.

4. PSEDUOCODE-SINGLE IMPUATATION USING GRNN

Step 1 : Normalize each variable to the range [0,1] using min- max normalization method.

"Min-max normalization subtracts the minimum value of a variable from each value of the variable and then divides the difference by the range of the attribute. This main reason for scaling the data to this range is to prevent the inputs with much larger ranges from dominating in the training." [1]

Step 2 : Design and construction of prototype model-P:

There are a set of n p-models(p_1,p_2,p_3,….p_n) for each missing value which is a single ensemble model(SEM_m) that estimates the conditional mean of a missing value. We also define a Indicator Matrix for each variable in X; $[I_{ij}]$=1 if the X_{ij} is missing and 0 if X_{ij} has a value.

The proposed algorithm is an iterative algorithm that consist of two steps: M-step and E-step. In M-step algorithm creates multiple models(one model for each of the missing value) and missing values are computed in the E-step

Step 2.1 : The maximization M-step

Create a loop to build a SEM_m model for each missing value

k=1; n=total

Step 2.1.1 : Relieff Algorithm to select 10 best feature subset for each missing value

Step 2.1.2 : Train a separate GRNN with the feature subsets for each missing value

k=k+1;

End(while k<=n loop)

Step 2.2 : The expectation E-step

E-step supplies imputed values based on imputation model calculated in previous step

Given a complete sample the next M-step updates the model P=(p_1,p_2,p_3…..p_n) and then E-step re-estimates the missing values based on the update. The two steps are iterated until the conditional means of missing values become stable. [1]

5. BLOCK DIAGRAM FOR GESI

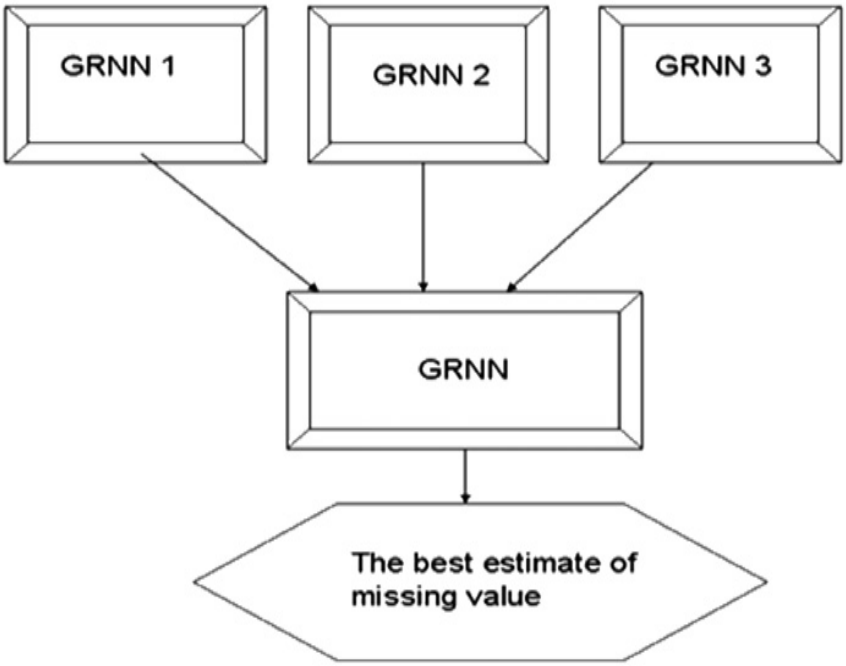


Figure 2 : Imputation model built by GESI. [1]

6. TOOLS USED

6.1 SOFTWARE : MATLAB

6.2 TOOLBOXES

1. Deep Learning Toolbox
2. DSP System Toolbox
3. Signal Processing Toolbox
4. Statistics and Machine Learning Toolbox

6.3 FUNCTIONS

1. newgrnn
   Used to design generalised regression neural network [4]
   Syntax : `net = newgrnn(P,T,spread)`

P	R-by-Q matrix of Q input vectors
T	S-by-Q matrix of Q target class vectors

`spread`	Spread of radial basis functions (default = 1.0)

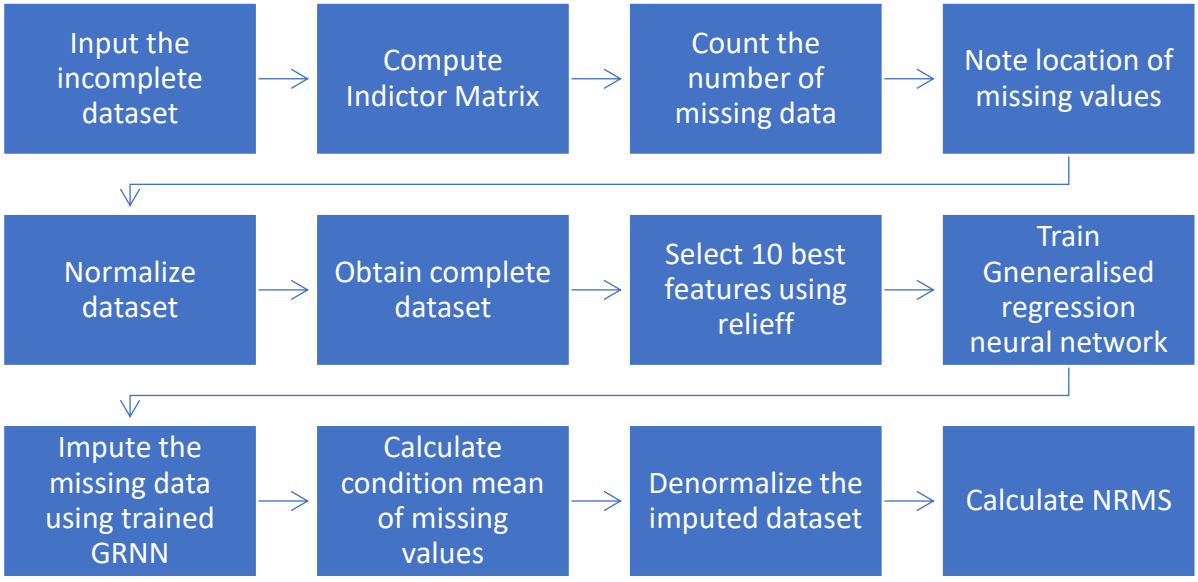
2. normalize
   Normalize dataset in the range of 0 to 1 [5]
   Syntax : `N = normalize(dataset,'range')`

3. rmmissing
   Remove rows consisting of missing entries [6]
   Syntax : `R = rmmissing(A)`

4. relieff
5. Rank importance of features [7]
   Syntax : `[Feature, Weight]=relieff(Input, Output,nearest neighbour)`

6. sim
   Simulate a neural network [8]
   Syntax : `Output_missing =sim(grnn,Input_missing)`

7. IMPLEMENTATION PROCEDURE



1. Input Incomplete dataset into MATLAB

1	2	3	4
	0	0.99539	-0.05889
1	2	1	-0.18829

1	0	1	-0.03365
		1	
1	0		-0.02401

2. Compute the indicator matrix

   Syntax: `indicator_matrix=isnan(dataset);`

1	2	3	4
1	0	0	0
0	0	0	0
0	0	0	0
1	1	0	1
0	0	1	0

3. Calculate no of missing values

   Syntax: `n=sum(indicator_matrix,'all');`

4. Normalise dataset using $(X_{ij}-X_{min})/(X_{max}-X_{min})$

1	2	3	4
NaN	0	0.99539	0.687238
1	1	1	1.11E-16
1	0	1	0.821286
NaN	NaN	1	NaN
1	0	NaN	0.872484

5. Find Location of each missing data

   Syntax: `[row,col]=find(indicator_matrix==1);`

6. Loop to iterating imputations of missing values

We stop iterating between expectation and maximization until difference between conditional means or successive iterations is less than 0.0001

Syntax: while difference(q)>0.0001

.

.

End

6.1. Obtain the complete dataset from the normalised matrix

Syntax: `rmmissing(Normalized_dataset);`

1	2	3	4
1	1	1	1.11E-16
1	0	1	0.821286

6.2. Iterating for n=no of missing values

6.2.1. Checking if there is any other missing value in the row (in which current missing value is being processed)

6.2.2. Extracting the output column for training the GRNN. (Output column is the column of missing data)

1
1
1

6.2.3. Extracting the Input variables for training the GRNN

6.2.3.1. Apply Relieff Algorithm to 10 features from the pool of features

2	3	4
1	1	1.11E-16
0	1	0.821286

6.2.4. Train the GRNN

Syntax: `newgrnn(Input',Output');`

6.2.5. Select Input for simulation

2	3	4
0	0.99539	0.687238

6.2.6. Imputing the missing values using trained GRNN

Syntax: `sim(Grnn,Input_sim');`

6.3. Building the imputed matrix. Assign imputed values to the missing values

```
for p=1: n
        Norm_dataset(row(p), col(p))=imputed(p);
End
```

6.4. Calculating conditional mean od missing values and difference between successive conditional means

7. De-normalise the matrix to obtain original matrix with imputed values

Syntax

```
for c=1: size(dataset,2)
   for r=1: size(dataset,1)
       minVal=min(dataset(:c));
       maxVal=max(dataset(:c));
       imputed_dataset(r, c)=minVal +
       (Normalized_dataset(r,c))*(maxVal-minVal);
   end
end
```

8. SUGGESTIONS FOR IMPROVEMENT

1. The algorithm doesn't work if the complete matrix(obtained after removing rows with missing value) is empty. Moreover, the datasets which have less than 20 complete rows produce results with high NRMS. A suggestion to improve the same would be to fill up the missing values initially with nearest neighbours using MATLAB function `fillmissing(Dataset,'nearest')`

2. The algorithm becomes computationally expensive for very large datasets as it involves computation of a large number of elements. Hence the report suggests to use transpose of the dataset and then select 500 optimal Instances for computation of each missing value.

3. Multiple imputation can be utilised to fasten the results

4. Use of cloud server(Google compute Engine/Amazon web services/Microsoft Azure,etc) can increase the computational speeds by 1000 times. Hence, a large dataset of Letter (20k rows) shall take 1 minute to process.

9. RESULTS

The performance of the algorithm can be evaluated by computing the NRMS Value

$$NRMS = \frac{||Xestimate\ - Xoriginal||F}{\big||Xoriginal|\big|F}$$

$$\big||A|\big|F = \sqrt{\sum_{i=1}^{m}\sum_{i=1}^{m}|aij|2}$$

The following results for NRMS Values are obtained

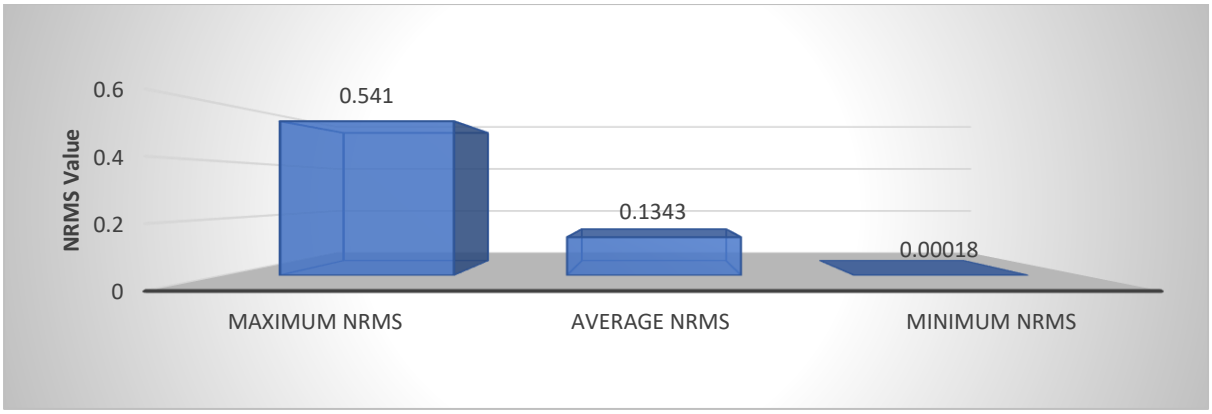


Figure 3 : Comparison of Graphical representation of NRMS Values achieved using the algorithm

The NRMS values obtained for different datasets are as below:

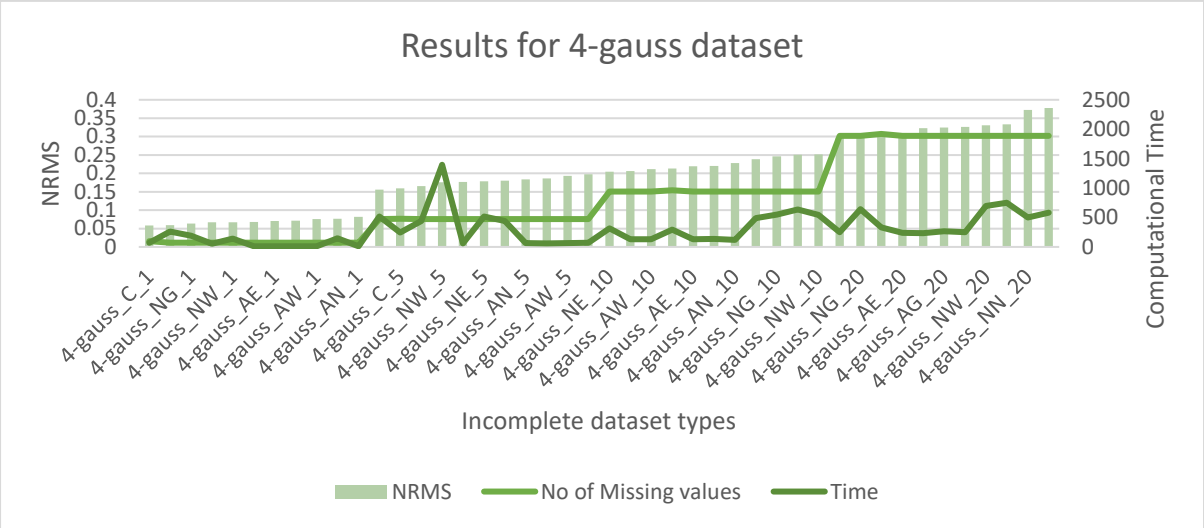


Figure 4 : Graphical representation of NRMS Values for 4-gauss dataset

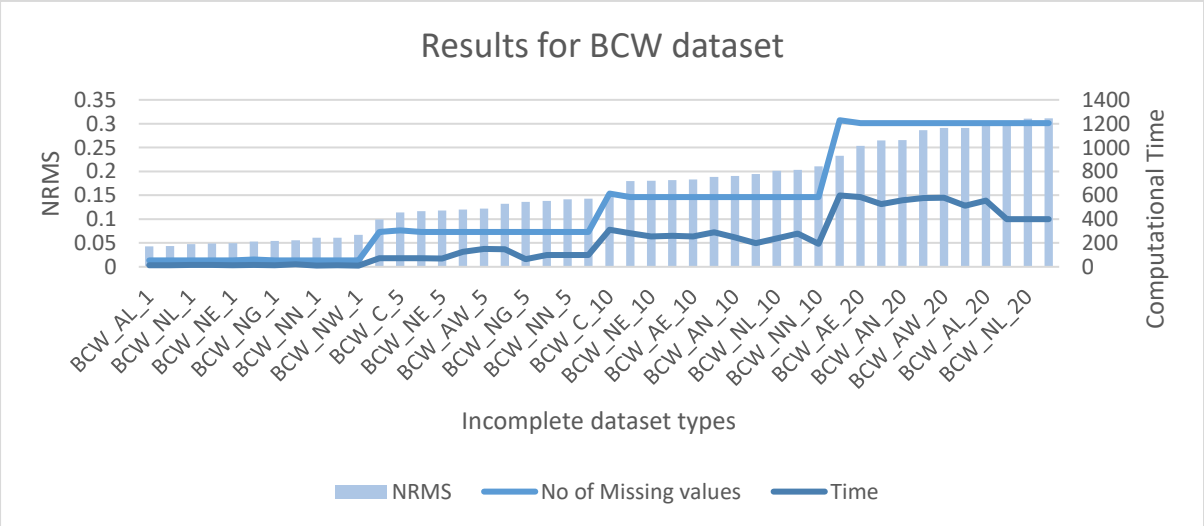


Figure 5 : Graphical representation of NRMS Values for BCW dataset

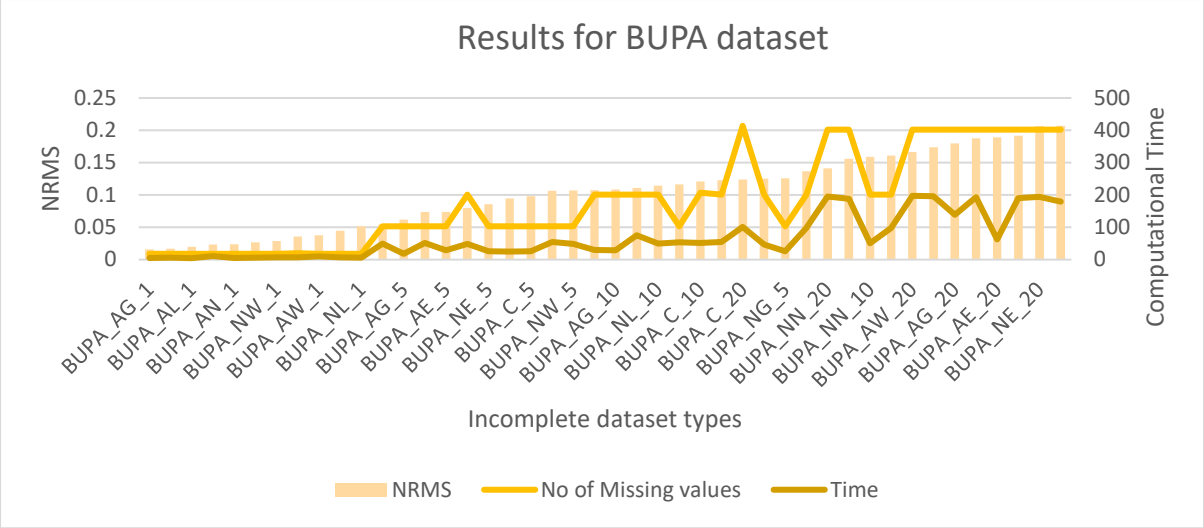


Figure 6 : NRMS Values for BUPA dataset

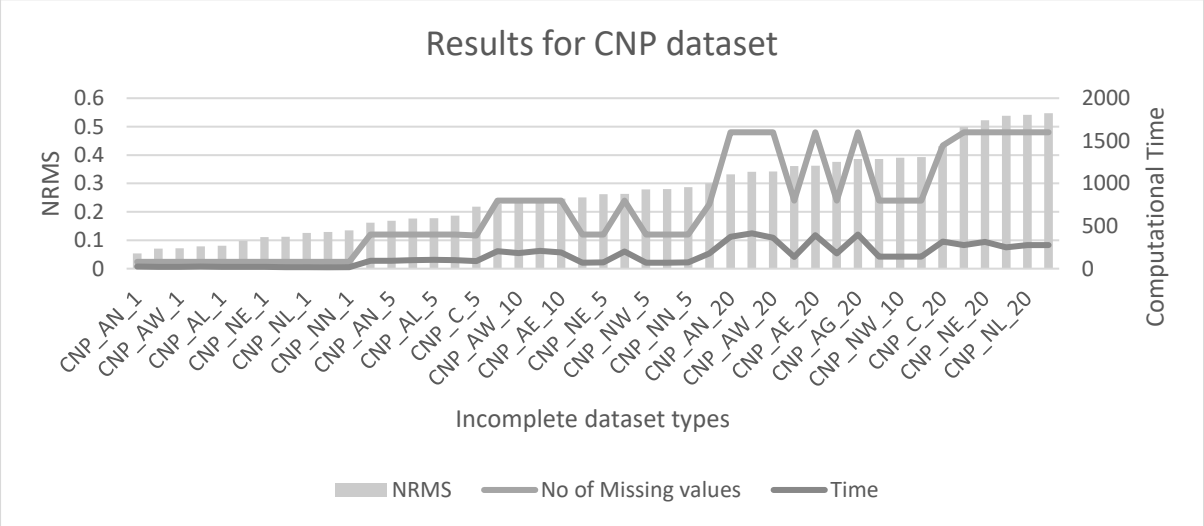


Figure 7 : Graphical representation of NRMS Values for CNP dataset

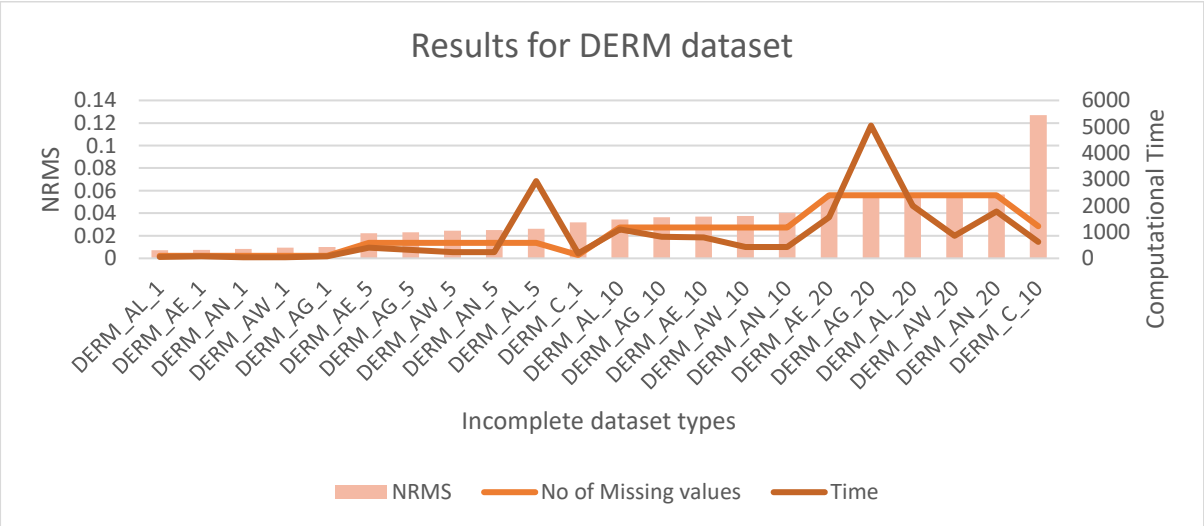


Figure 8 : Graphical representation of NRMS Values for DERM dataset

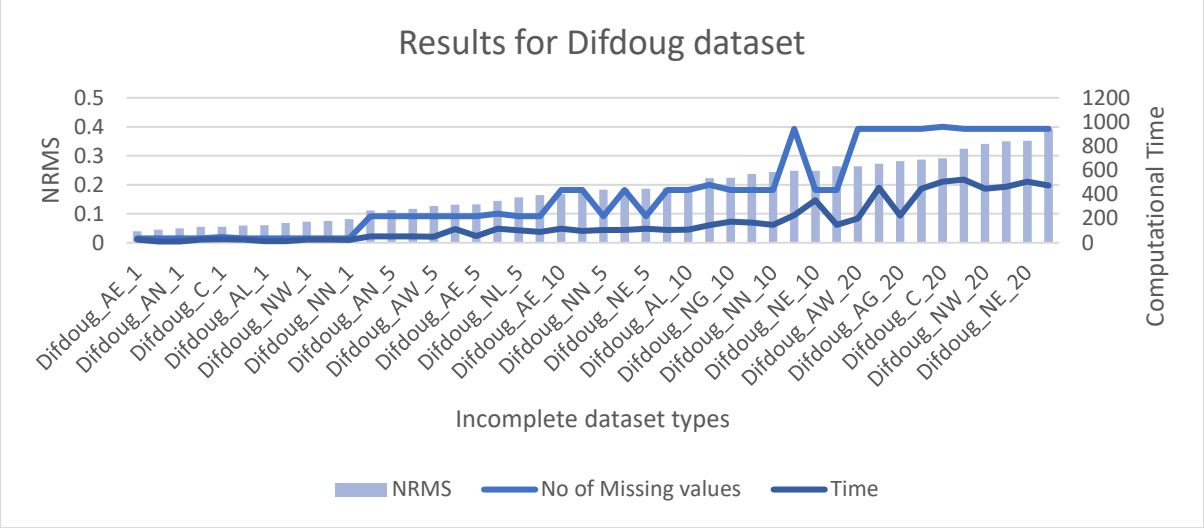


Figure 9 : Graphical representation of NRMS Values for Difdoug dataset

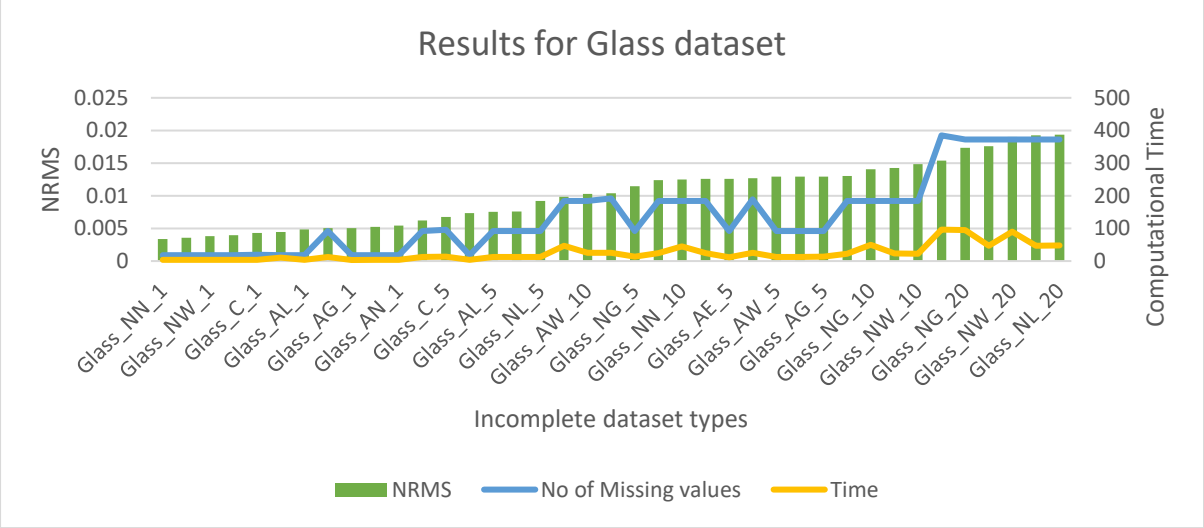


Figure 10 : Graphical representation of NRMS Values for Glass dataset

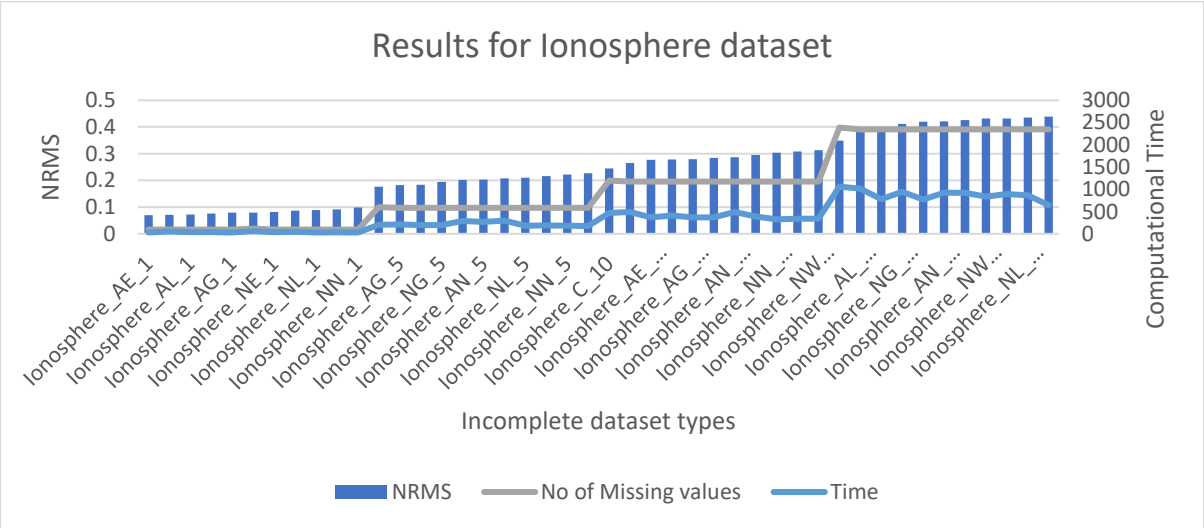


Figure 11 : Graphical representation of NRMS Values for Ionosphere dataset

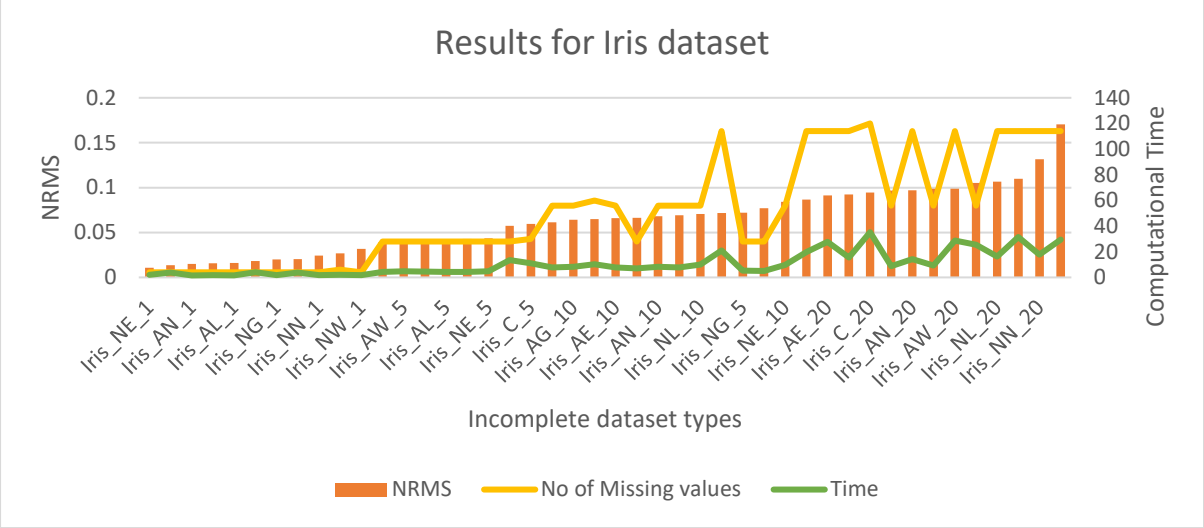


Figure 12 : Graphical representation of NRMS Values for Iris dataset

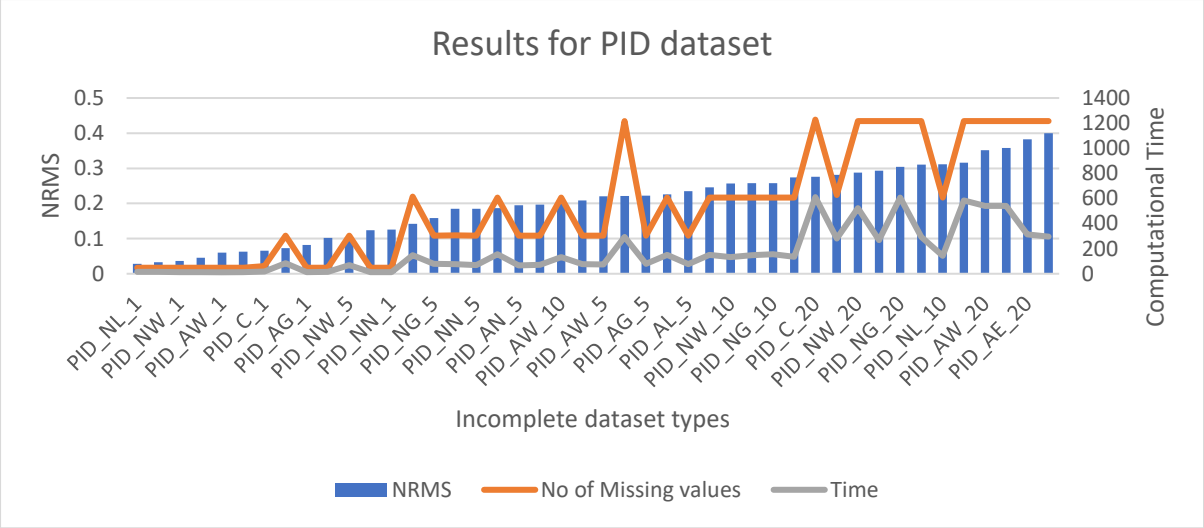


Figure 13 : Graphical representation of NRMS Values for PID dataset

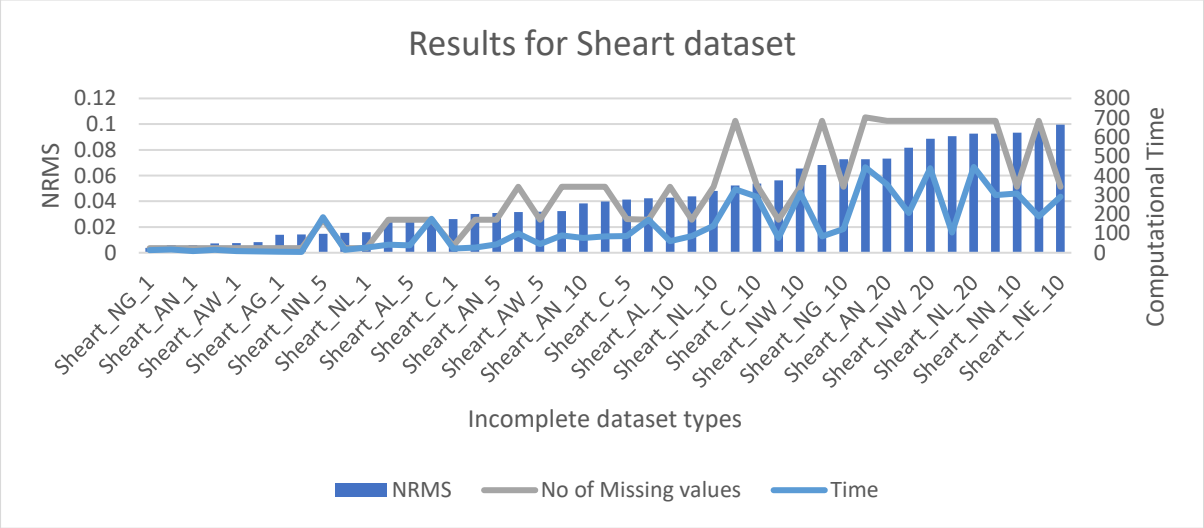


Figure 14 : Graphical representation of NRMS Values for Sheart dataset

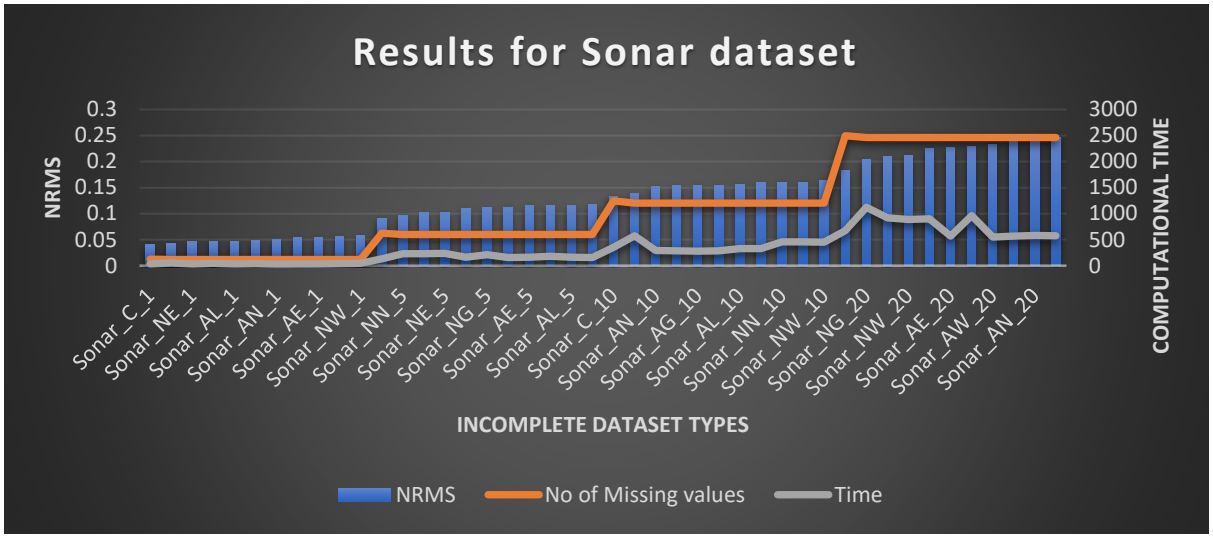


Figure 15 : Graphical representation of NRMS Values for Sonar dataset

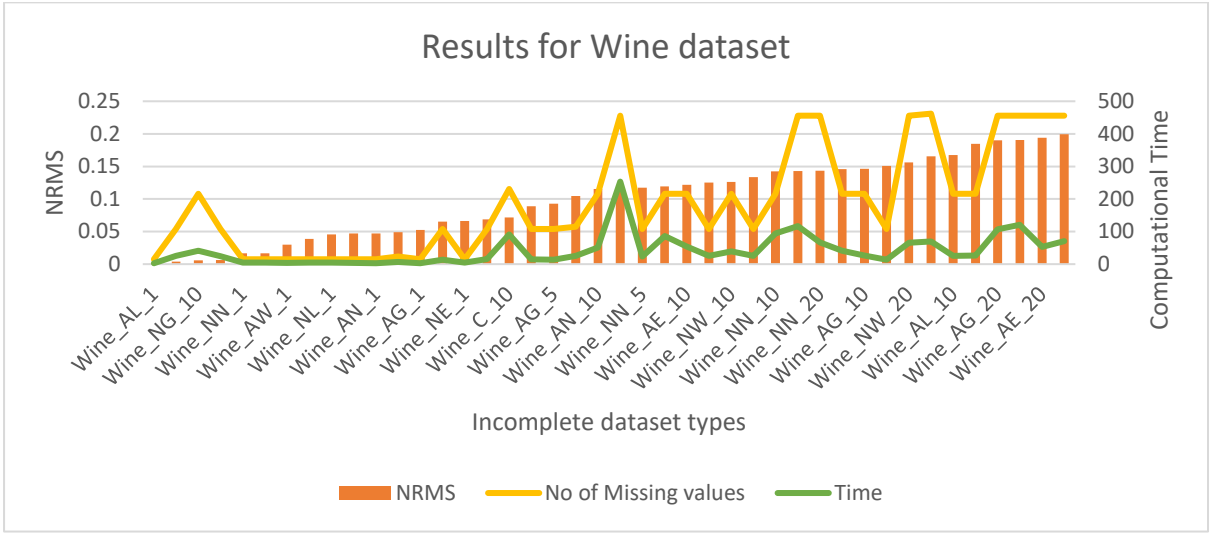


Figure 16 : Graphical representation of NRMS Values for Wine dataset

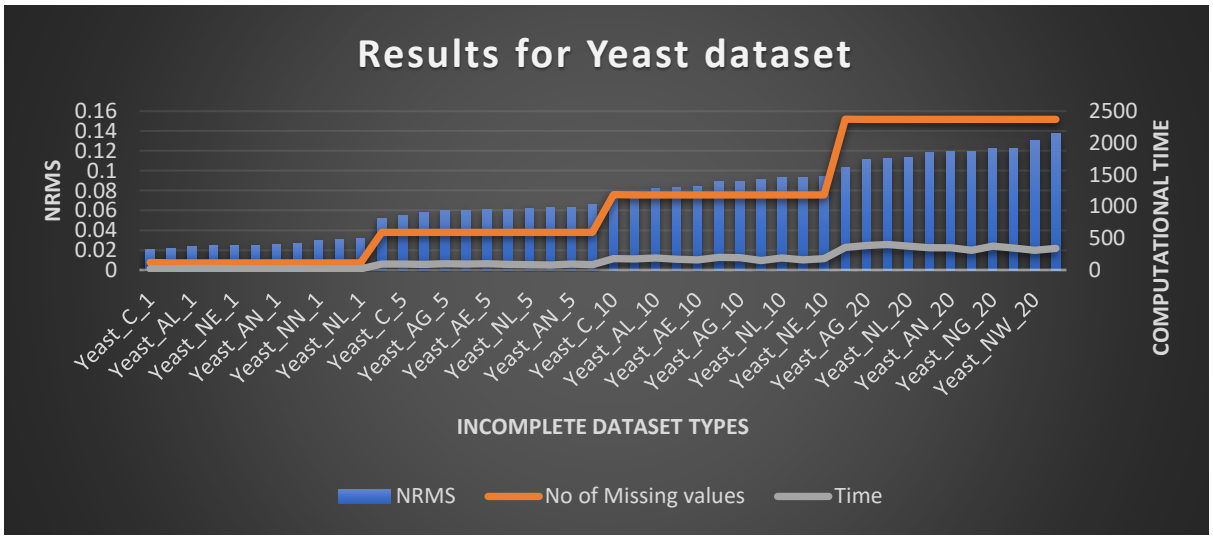


Figure 17 : Graphical representation of NRMS Values for Yeast dataset

10. CONCLUSION

The report has presented Single imputation algorithm GESI which uses generalized regression neural network. The algorithm has been tested on types of 12 datasets having 44 variations in terms of location and number of missing values. Hence , a total of 528 datasets have been processed using the algorithm and MATLAB code. Results for large datasets like Letter were computationally expensive, hence not obtained. However, we propose to use cloud cluster parallel computing to enhance the processing speed by 1000 times.

Interesting highlights

1. The highest NRMS Value are obtained for CNP datasets with missing rate of 20%, since CNP dataset has only two features for compute. This showcases that having more number of features increases the computational accuracy.
2. Wine and glass datasets with missing rate of 1% -5% have extremely good NRMS values (less than 0.003).
3. Iris is the fastest data that was computed in 1.6 seconds whereas the DERM dataset took 1.4 hrs
4. Sonar datasets with missing rate of 20% had the highest number of missing values, however the computational time was 11 minutes only and an acceptable missing rate(less than 0.2).
5. The Relieff feature selection algorithm is the most time intensive. Hence faster and better feature selection algorithms could be used
6. The results can be improved further if the proposed algorithm is used in series with another missing value imputation algorithm. The NRMS values obtained after filling the dataset with nearest neighbour was 0.8 which was reduced to 0.2 by processing the results with GESI

REFRENCES

[1] L. Iffat A.Gheyas n, "A neural network-based frame work for the reconstruction of incomplete data sets," ElsevierB.V., Stirling,Scotland, 2010.

[2] gopinath.jgec, "Basic Neural Network : Algorithm and Example," [Online]. Available: https://easyneuralnetwork.blogspot.com/2013/07/grnn-generalized-regression-neural.html. [Accessed 01 06 2019].

[3] MATLAB, "Deep Learning Toolbox," MATLAB Inc., [Online]. Available: https://www.mathworks.com/help/deeplearning/. [Accessed 05 06 2019].

[4] "newgrnn," MATLAB, [Online]. Available: https://www.mathworks.com/help/deeplearning/ref/newgrnn.html?s_tid=doc_ta. [Accessed 05 06 2018].

[5] "normalize," MATLAB, [Online]. Available: https://www.mathworks.com/help/matlab/ref/double.normalize.html?s_tid=doc_ta. [Accessed 05 06 2018].

[6] "rmmissing," MATLAB, [Online]. Available: https://www.mathworks.com/help/matlab/ref/rmmissing.html?s_tid=doc_ta. [Accessed 06 06 2019].

[7] "relieff," MATLAB, [Online]. Available: https://www.mathworks.com/help/stats/relieff.html?s_tid=doc_ta. [Accessed 07 06 2019].

[8] "sim," MATLAB, [Online]. Available: https://www.mathworks.com/help/deeplearning/ref/sim.html?s_tid=doc_ta. [Accessed 05 08 2019].