UNIVERSITY OF CALIFORNIA
Los Angeles

Trees vs Neurons:
Comparison between Denoising Autoencoders and Random Forest for Imputation of Mixed Data from Electronic Medical Records

A thesis submitted in partial satisfaction
of the requirements for the degree
Master of Applied Statistics

by

Qin Peng

2018

ABSTRACT OF THE THESIS

Trees vs Neurons:
Comparison between Denoising Autoencoders and Random Forest
for Imputation of Mixed Data
from Electronic Medical Records

by

Qin Peng

Master of Applied Statistics
University of California, Los Angeles, 2018
Professor Ying Nian Wu, Chair

Missing data is a significant challenge impacting almost all studies, especially for the analyses of electronic health record (EHR). We propose a multiple imputation model based on multi-layer denoising autoencoders. This nonparametric model can deal with mixed-type data without making assumptions about the missing mechanism. Evaluation of simulated datasets based on real-life EHR datasets showed that our proposed model outperforms both the Random Forest imputation method and the median/mode imputation method.

Keywords: Denoising autoencoder, imputation, electronic health records (EHR), mixed-typed data, nonparametric model

The thesis of Qin Peng is approved.

Nicolas Christou

Qing Zhou

Ying Nian Wu, Committee Chair

University of California, Los Angeles

2018

To my mother, my father, and my husband...

who—among so many other things—

saw to it that I learned to touch-type

while I was still in elementary school

Table of Contents

List of Figures

List of Tables

ACKNOWLEDGMENTS

Foremost, I would like to express my sincere gratitude to my advisor Prof. Ying Nian Wu for his continuous support of my study and research, and for his patience, motivation, enthusiasm and immense knowledge. His guidance helped me in all the time of my research and the writing of this thesis. I could not have imagined having a better advisor and mentor for my Masters' study.

Besides my advisor, I would like to thank the rest of my thesis committee: Prof. Qing Zhou and Prof. Nicolas Christou, for their encouragement, insightful comments, and hard questions.

Last but not the least, I must express my very profound gratitude to my parents and to my husband for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

Introduction

1.1 Background

Data quality is of great significance and has been continuously concerned in the Machine Learning and other data analysis area. Most mainstream Machine Learning algorithms use existing datasets to build and train models, then use models to predict further dataset. The quality of the models highly depends on the quality of the data[Bre]. Bad input data could result in the bad quality of models. In the worse case, the dataset could even contain missing data. Missing data could potentially bias statistical analyses, resulting in misleading conclusions. For Machine Learning algorithms, the presence of missing value could even cause more serious problems because most of the Machine Learning algorithm will simply crash if presenting datasets that consist of missing data[GR15]. In some special cases, it might be possible to retrieve original value(s) of the missing data[SE17], but unfortunately, in most real-life cases, it is unlikely to acquire them[HKS17]. Though scholars and scientists have proposed various imputation methods to deal with missing data, which could predict missing values in the datasets based on existing values and other knowledge, there is no golden rule among these methods and different imputation methods may work for different situations.

1.2 Purpose and Rationale

In this thesis, we will be discussing and comparing different kinds of missing data imputation methods that could be applied to Electronic health record (EHR) data.

Electronic health record (EHR) is defined as a collection of patients' paper charts in the digital version[YSK18]. Scientists have been using EHR to build statistical and Machine Learning models, which can be used to predict different clinical information such as patient trajectory[ESG], cancer prediction[ZW11], etc.

Because EHR systems are designed for clinical purposes, the quality of EHR data is not always good. Missing data is quite common in EHR data. Moreover, the clinic environment also contributes to the missing data[LLK14]. For example, a unhealthy financial status of patients could lead to missing some lab tests and/or medical treatments because these tests and treatments may not affordable for these patients.

1.3 Basic concepts

1.3.1 Electronic health records(EHR)

Electronic health records (EHRs) consist of a patient's information: demographics, timeline, chief complaint, laboratory test results, diagnosis, medical procedures applied including medication directions, etc[YSK18].

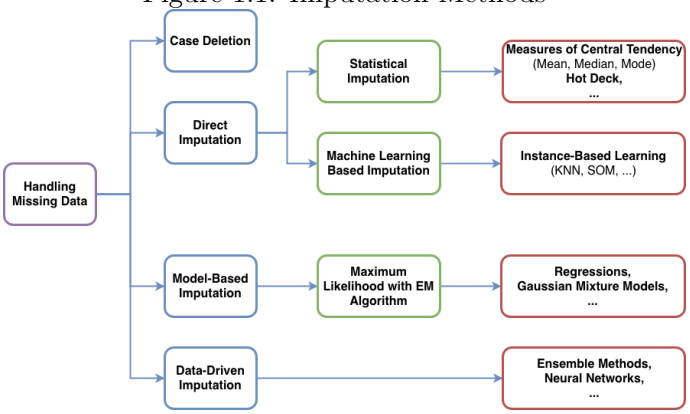
In the mid-1960s, Lockheed developed the world's first electronic health record system known as a clinical information system. On February 17, 2009, the HITECH Act, as a part of the American Recovery and Reinvestment Act, was signed into law. As the act encouraged health care providers and hospitals to adopt EHR technology at a greater distance[Sto], according to the latest report from the Office of the National Coordinator for Health Information Technology (ONC), Basic or more advanced EHR system has been adopted in approximately 84% of hospitals in the United States, which is a nine-time increase comparing to the number in 2008[STB18].

1.3.2 Why data-driven models

> There are two cultures in the use of statistical modeling to reach conclusions from data. One assumes that the data are generated by a given stochastic data models. The other uses algorithmic models and treats the data mechanism as unknown...Algorithmic modeling, both in theory and practice, has developed rapidly in fields outside statistics. It can be used both on large complex data sets and as a more accurate and informative alternative to data modeling on smaller data sets. If our goal as a field is to use data to solve problems, then we need to move away from exclusive dependence on data models and adopt a more diverse set of tools.
>
> —Leo Breiman[Bre]

Scientists have been working on dealing with missing data for a long period of time and have proposed various methods to deal with them. A summary of the methods is shown in Figure 1.1. Some of these methods have been gaining a resurgence in recent years, which are essentially powerful when dealing with data from clinical trials and in biomedical studies. However, all these methods are associated with various drawbacks when used for data mining. It is essential for model users to minimize model bias, as well as avoid underestimating or overestimating model variability.

Figure 1.1: Imputation Methods



1.3.3 Model-Driven Approaches

When we have a deep understanding of the system or process of interest, and we know what is the suitable model(s) to choose, model-driven approaches are powerful and scientifically beautiful. However, the reality is so complex, it is nearly impossible for us to use a model without abstraction and simplification, which may harm the model generalizability. Furthermore, choosing the best model and features takes time. The trial-and-error process of finding a suitable model and refining it is often a lengthy process.

1.3.4 Data-Driven Approaches

Instead of making explicit assumptions, representation, and rules, data-driven approaches based on machine learning often require a fairly large amount of data, and the more data a model is fed, the better results one could get. It frees researchers and analysts from the burden of guessing the underneath mechanisms of data. Data-driven learning encourages more evidence to be brought forward and knowledge to be updated. However, data-driven is data hungry and often computational consuming[WSH18].

1.4 Missing Data Mechanisms

1.4.1 Nonresponse Mechanisms

The response here means both data collected directly from respondents, and data collected from an administrative process. This broad definition of response is necessary to reflect the increased use of different collection strategies in the same survey, a practice that has become more and more commonplace.

Researchers, survey managers, and operations staff tried hard to maximize response. However, most, if not all surveys still have to deal with nonresponse. Different reasons can cause nonresponse. In some cases, users did not respond to questions in a recommender system, and in some other cases patients simply pass away and will no longer be able to answer questions. There could even be internal or infrastructure bugs or errors in the database that stores data, and so on.

Nonresponse can have two effects on data: first, it introduces a bias in estimates when nonrespondents differ from respondents in the characteristics measured; second, it contributes to an increase in the total variance of estimates since the sample size observed is reduced from that originally sought.

There are two types of nonresponse.

1.4.1.1 Unit Nonresponse

If none or only very limited information known about a unit/case, we regard it as unit nonresponse. For example, if a patient recently registered with a practice, we know their address, date of birth and sex, but if the patient did not consult, no health indicator measurements would be recorded.

1.4.1.2 Item Nonresponse

Item Nonresponse is incomplete case data of which the degree of item response meets a minimum threshold. For example, a patient could have their weight and/r blood pressure recorded at regular time intervals, but one could rarely be recording total cholesterol measurements at that pace.

In this study, we only focus on the item non-response.

1.4.2 Missing Mechanisms

1.4.2.1 Missing completely at random (MCAR)

If a value in a dataset is missing completely at random (MCAR), that means the event caused the particular item being missing is independent of not only all the observable variables but also all unobservable parameters related. Furthermore, the event occurs entirely at random. If data are MCAR, the missingness of data will be unrelated to any study variable, thus the analysis performed on the dataset will be unbiased. However, MCAR rarely happens.

1.4.2.2 Missing at random (MAR)

Comparing to MCAR, Missing at Random (MAR) is a weaker assumption. It assumes that although the missingness is not random, whether an observation is missing will not be based on the missing values. The missing values only have a systematic relationship with the observed data.

Many widespread imputation software packages, either using maximum likelihood[Hip12] or multiple imputations[All], depend on the assumption of MAR. However, MAR it's still hard to be verified statistically, there will be always possible that we have unobserved variables contributing to the missingness. Unlike MCAR, MAR data may induce parameter bias.

However, if we estimate a parameter with full information maximum likelihood, we can get asymptotically unbiased estimates from MAR[EB01].

1.4.3 Missing not at random (MNAR)

MNAR stands for Missing Not at Random, it is also known as non-ignorable(NI). When a data is missing but is neither MCAR nor MAR, we call it MNAR. In that case, the value of the missing variable is not only related to the observed data but also deal with the missing part itself.

1.5 Imputation approaches

In this study, we imputed missing data in the EHR by three methods: the simple median/mode imputation; the multiple imputation by denoising autoencoder; and random forest using missRanger package in R. Firstly, We spiked-in missing data to the Province dataset and evaluated each approach's imputation performance. Each of these is described in detail below and except MIDA, all analysis was run using free open source library packages, missRangerR, Tensorflow, and Scikit-learn.

1.5.1 Simple Imputation

Instead of simply dropping the data with missing values, the most common and straightforward imputation method is to impute a plausible value for the missing observations, such as for continuous variables, impute by filling in the mean; for categorical variables, impute by filling in the mode.

1.5.2 Random forest Imputation

1.5.2.1 missForest

Stekhoven et al.[SB12] developed a random forest-based algorithm called missForest that can be used for missing data imputation. This algorithm focuses on predicting accurate values for each individual missing values, while other algorithms may result in a distribution and take random draws as predicted values.

MissForest directly predicted the missing values using a random forest trained on the observed parts of the data set. It is a nonparametric imputation method which can be used to address mixed-type of variables and build nonlinear relations. It also supports high dimensional data. For each variable, missForest fits a random forest on the observed part and

then predicts the missing part. The pseudo algorithm is provided below:[SB12]

Figure 1.2: Pseudo algorithm for missForest

Algorithm 1 Impute missing values with random forest.

Require: $\mathbf{X}$ an $n \times p$ matrix, stopping criterion γ

1: Make initial guess for missing values;
2: $\mathbf{k} \leftarrow$ vector of sorted indices of columns in $\mathbf{X}$
   w.r.t. increasing amount of missing values;
3: **while** not γ **do**
4: $\quad \mathbf{X}_{old}^{imp} \leftarrow$ store previously imputed matrix;
5: $\quad$ **for** s in $\mathbf{k}$ **do**
6: $\quad\quad$ Fit a random forest: $\mathbf{y}_{obs}^{(s)} \sim \mathbf{x}_{obs}^{(s)}$;
7: $\quad\quad$ Predict $\mathbf{y}_{mis}^{(s)}$ using $\mathbf{x}_{mis}^{(s)}$;
8: $\quad\quad$ $\mathbf{X}_{new}^{imp} \leftarrow$ update imputed matrix, using predicted $\mathbf{y}_{mis}^{(s)}$;
9: $\quad$ **end for**
10: $\quad$ update γ.
11: **end while**
12: **return** the imputed matrix $\mathbf{X}^{imp}$

1.5.2.2 missRanger

We first decided to use missForest for our Random Forest Imputation, but due to the large data size, it kept running for days without giving out any output, even after parallelizing the missForest function. Thus, we moved on to another R package called "missRanger". From the documentation, missRanger is an alternative implementation of 'MissForest' algorithm used to impute mixed-type datasets by chaining tree ensembles[WZ17].

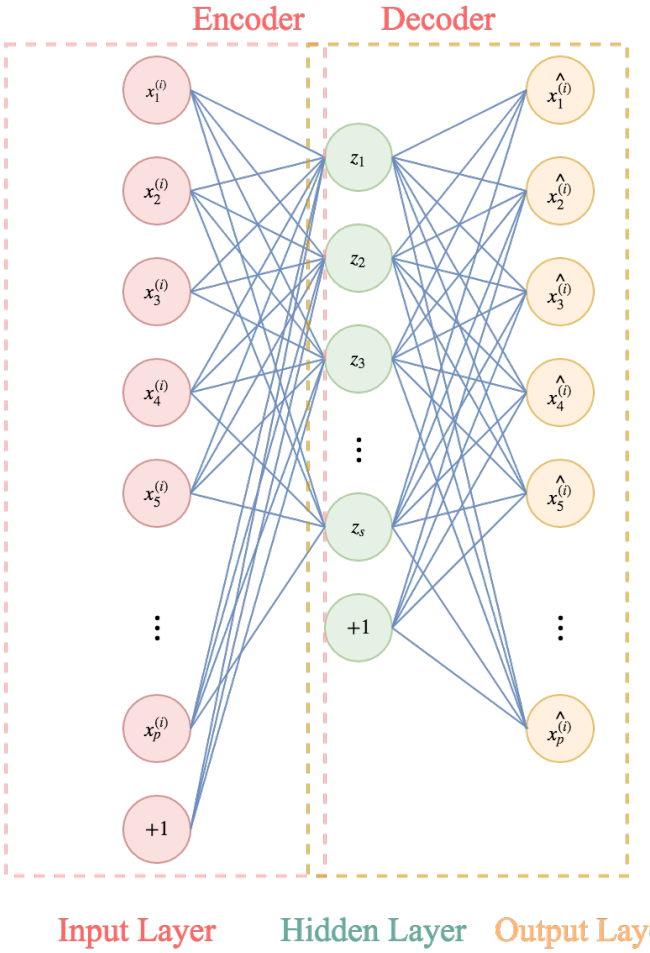
1.5.3 Denoising autoencoders

1.5.3.1 Autoencoder

An autoencoders is a unsupervised neural networks that reconstruct their outputs from their input without remembering trivial details[JTS17].

In its simplest form, the Vanilla Autoencoder only contains one input layer, one hidden layer, and one output layer.

Figure 1.3: Vanilla Autoencoder



An autoencoder takes an input $x \in [0,1]^d$ and encodes it to a latent representation $z \in [0,1]^d$ using an encoder, where d' represents a different dimensional subspace. The assumption is, in the dataset, z captures the coordinates along the main factors of variation. Then we decoded the encoded representation back to the original d dimensional space using a decoder. Encoder and decoder are both neural networks. The two stages are represented as

$$z = s(Wx + b)$$

$$o = s(W'x + b')$$

where o is the decoded result and s is a nonlinear activation function. In the model training, we minimized the reconstruction error between x and o.

By increasing the number of hidden layers and using non-linear activation functions for each layer, the embedding can learn to encode complex, higher-level features such as pictures. Our MIDA model adopted the multi-layer version.

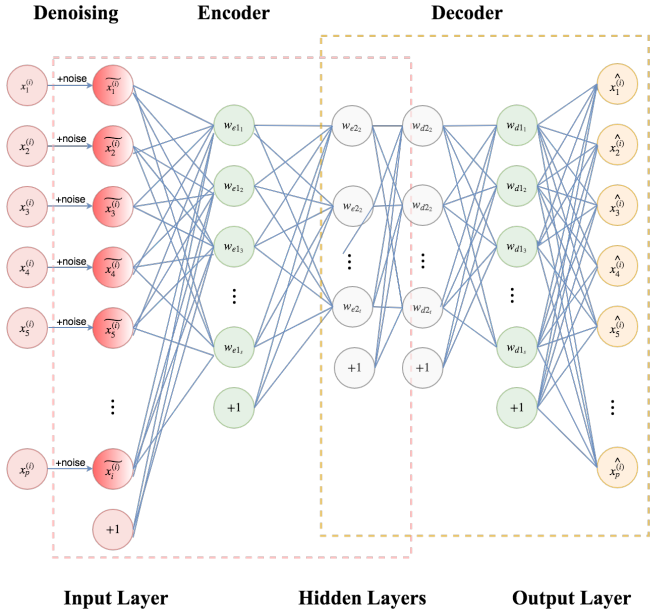
Denoising autoencoders(DAE) are an extension to autoencoders. By corrupting the input data x with noise to obtain $\widetilde{X}$ and forcing the network to reconstruct the clean output forces the hidden layers to learn robust features. The DAE is trained to reconstruct the original, noise-free input X from $\widetilde{X}$. Corruption can be applied in dierent ways. In this study, the noise is added to a random selection of X with three common forms: a) Gaussian noise; b) masking noise, where elements of X are set to 0; or c) salt and pepper noise, where elements of X are set to their minimum or maximum value with the same probability. DAEs reconstruction capabilities can be explained by thinking of DAEs implicitly estimating the data distribution as the asymptotic distribution of the Markov chain that alternates between corruption and denoising.

MIDA(Multiple Imputation by Denoising Autoencoder)

2.1 Architecture

Our default architecture is shown below.

Figure 2.1: Diagram of MIDA Model



2.1.1 Preprocessing Input

The first step is preprocessing input to fit our model. We rearranged the input dataset with all numerical variables at first, followed by each One-Hot encoded categorical variables. Then we applied sklearn.preprocessing.MinMaxScaler( ) to standardize our input between 0 and 1 to facilitate faster convergence. We choose MinMaxScaler because it will not affect the One-Hot encoded values, while other standardizers might.

2.1.2 Adding Noises to Input

Then we added noises to our input. We provide three kinds of denoising methods: 1) masking noise, which randomly masks values as zero by given rate; 2) salt and pepper noise, which a given ratio of elements of input is set to its maximum or minimum value according to a fair coin flip; and 3) Gaussian noise, adding Gaussian noise with mean equals 0, standard deviation equals given sigma to data in input.

2.1.3 Building Autoencoder Model

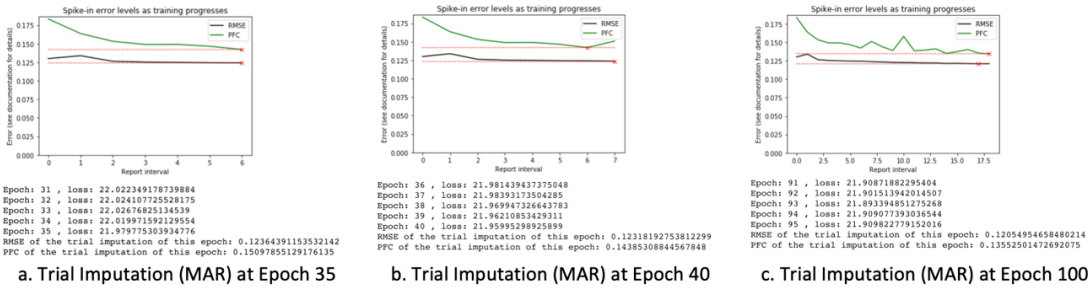
Next, we employ symmetrical Multilayer perceptron(MLP) as encoder and decoder. Instead of ReLU, we use ELU(Exponential Linear Unit) to avoid the vanishing gradient problem but also tend to converge cost to zero faster and produce more accurate results. This model building step involves two mini-steps:

2.1.3.1 Trial Imputation for Hyperparameter Choosing

We sampled a subset of complete data in the input, then randomly spiked in missing values. We used the sampled subsets as training and testing dataset. By examining the visualized RMSE and PFC plots, we decided our hyperparameters. In this study, we used a three-layer encoder with hidden nodes of 512, 256, 256 respectively, and a reversed decoder with hidden

nodes of 256, 256, 512. It is an arbitrary choice. I chose 512 because after preprocessing, our input has 264 features, and 512 is the closest power-of-two larger than it. Same as Gondara and Wang's work[GW17], due to the large degree of regularization, making a model "too big" is less of a problem than making it "too small".

Figure 2.2: Sample Trial Imputation Visualization



a. Trial Imputation (MAR) at Epoch 35 b. Trial Imputation (MAR) at Epoch 40 c. Trial Imputation (MAR) at Epoch 100

Our model is trained around 50 epochs using an adaptive learning rate with a time decay factor of 0.99. The Figure 2.2 showed RMSE and PFC along epochs in MAC trial imputation. We see epoch 35 is the first inflection point for PFC. As the trial imputation going on, the trial loss and trial RMSE decreased slightly, while PFC presented as a zigzag line. Thus, for MAC imputation, we choose training epoch for 35.

2.1.3.2 Model training

After choosing hyperparameters, we trained the autoencoder with the whole dataset.

2.1.4 Output Imputation

We used the trained model to generate Imputed dataset 20 times.

2.1.4.1 Point Estimates

Our MIDA package provides point estimates. For numerical variables, MIDA will output mean and standard deviation of the multiple imputed data; for categorical variables, MIDA will output the mode and frequency of the value in the multiple imputed data. No point estimate performance comparison is done in this study because neither missRanger nor Simple Imputation has a similar function. A demonstration is offered in Github.

2.1.4.2 Whole Dataset Imputation

Our MIDA package also outputs the imputed dataset as a whole for use. Imputation performance comparison results are presented in the following section.

Experiment design

3.1 Data preparation and standardization

We compared different missing value imputation methods in both simulated data and real datasets. We evaluated the imputation performance by calculating root mean squared error ($RMSE$) for numerical variables and proportion of false classification (PFC) for categorical variables.

3.2 Missingness simulation

[BLS17] Missingness simulation is often used to generate various missing scenarios to test the performance of missing data imputation techniques. In this study, four missingness mechanisms (MMs) shall be simulated to generate 4 incomplete dataset versions. Missingness ratio (MR) of each variable is set to match the one in the simulated raw dataset. We adopted the simulation process in the Beaulieu-Jones etc.'s work[BLS17].

3.2.1 Simulation 1: Missing Completely at Random (MCAR)

We replaced values with NAN at random. Missing values are overall selected completely at random. Assume we have $MR_i = 5\%$ missing values of the $\mathrm{i_{th}}$ variable in the simulated dataset, around $0.05 * N$ missing values will be injected randomly inside the variable. N is the sample size of the simulated dataset.

3.2.2 Simulation 2: Missing at Random (MAR)

It is generated by making the distribution of missing values depends on the last two variables "READMITIN30DAYS" and "READMITIN90DAYS", which indicate whether the patient is readmitted within 30 and 90 days. We choose those anchor variables because they are complete, and the readmission rates are important indicators in healthcare reform and cost-containment studies.

For the i_{th} variable with the missing rate MR_i, we implement a biased selection process where $100\%\, MR_i$ missingness is injected into the RA00 instances, i.e. both "READMITIN30DAYS" and "READMITIN30DAYS" equal to 0. And $75\%\, MR_i$ missingness is injected into the RA01 instances who are not readmitted in 30 days but in 90 days. $120\%\, MR_i$ missingness is injected into the RA11 instances who are readmitted in both 30 and 90 days.

3.2.3 Simulation 3: Missing not at Random (MNAR)

It is generated by making the distribution of missing values not only depends on last two variables "READMITIN30DAYS" and "READMITIN90DAYS", but also depends on the value itself.

Derived from the simulation process in MAR, in which we inject $100\%MR_i$ missingness into RA00 instances, $75\%MR_i$ missingness into RA01 instances, and 120% missingness into RA11 instances, we further divided cases into the threshold group and non-threshold group by the procedure below:

First, we pick the threshold $thres_i$ for each variable, $i \in (1, 2, \cdots, 118)$.

For the i^{th} variable, if it is a categorical variable, the threshold $thres_i$ is the most frequent value, the threshold set for the i^{th} variable $Thres_i$ is all cases with the value equals to the threshold, and p_i is the percentage that the the value of this variable is the threshold, or the probability of which a case is in the i^{th} threshold set $Thres_i$.

$$p_i = \frac{\text{number of cases with value equals to threshold of the } i^{th} \text{ variable}}{\text{number of non-missing cases of the } i^{th} \text{ variable}}$$

If the i^{th} variable is a numerical variable, and if most values of the variable are the same, i.e., the 25^{th} percentile is the same as the 50^{th} percentile, or the 50^{th} percentile is the same as the 75^{th} percentile, the threshold $thres_i$ is also the most frequent value, the threshold set is again the cases with the value equals to the threshold, and p_i is the percentage that the the value of this variable is the threshold.

$$p_i = \frac{\text{number of cases with value equals to threshold of the } i^{th} \text{ variable}}{\text{number of non-missing cases of the } i^{th} \text{ variable}}$$

Otherwise, the threshold $thres_i$ is the 50^{th} percentile, the threshold set of the i^{th} variable is all the cases with value of the i^{th} variable no less than $thres_i$, p_i is the percentage that the the value of this variable is no less than the threshold.

$$p_i = \frac{\text{number of cases with values no less than the threshold of the } i^{th} \text{ variable}}{\text{number of non-missing cases of the } i^{th} \text{ variable}}$$

Second, for each variable V_i, if a case is in the threshold set $Thres_i$ with the probability of p_i, we need to calculate two parameters, a_{in} and a_{out} by the rules: If $p_i < 0.50$, then $a_{in} = 1.10$ and $a_{out} = (1 - 1.10 * p_i)/(1 - p_i)$. Else $a_{out} = 1.10$ and $a_{in} = (1 - 1.10 * (1 - p_i))/p_i$.

Then if the value is in the threshold set, we implement a biased selection process where $a_{in} \times MR_i$ random missingness in the cases not be readmitted either in 30 days or in 90 days; $a_{in} \times 0.75 \times MR_i$ in the cases not be readmitted either in 30 days but in 90 days; and $a_{in} \times 1.20 \times MR_i$ in the cases be readmitted both in 30 days an 90 days. If the value is not in the threshold set, we use the a_{out}.

3.2.4 Simulation 4: By Real data (KNN)

For real datasets, we first generated the complete dataset from the original raw dataset with missing values. From the complete cases dataset, we took each patient and matched them to the nearest neighbor, excluding self-matches, in the entire set of the raw dataset based on their age, time spent in ICU, and length of stay for the index admission. We then replaced any variable value with NAN if it was absent for the matched patient in the original data. Thus, we get our simulated dataset.

3.3 Evaluation Techniques

3.3.1 Categorical Variables

3.3.1.1 Accuracy

Accuracy is the number of correct predictions made divided by the total number of predictions made.

In this study, accuracy is calculated by sklearn.metrics.accuracy_score( )function.

3.3.1.2 F_1 Score

The F_1 Score is another measure of classification accuracy. In binary classification, it is $\frac{2 \times precision \times recall}{precision + recall}$. When we have perfect precision and recall, F_1 score equals 1, and worst value is 0. In the multi-class and multi-label case, F_1 Score is actually the average of the F_1 Score of each class with weighting.

In this study, F_1 Score is calculated by sklearn.metrics.f1_score( ) function.

3.3.2 Numerical Variables

3.3.2.1 Root Mean Square Error(RMSE)

The root-mean-square error (RMSE) is the square root of the differences between predicted values and observed values, or the quadratic mean of these differences. RMSE is always non-negative, and a value of 0 (almost never achieved in practice) would indicate a perfect fit to the data. In general, a lower RMSE is better than a higher one. RMSE is a commonly used measure of accuracy to compare test errors of different models for a given training dataset.

Because RMSE is scale-dependent, it is not suitable for between datasets comparison.

In this study, we used sklearn.metrics.mean_squared_error( ) function to calculate MSEs, then used numpy.sqrt( ) to get the square roots.

3.3.2.2 Median Absolute Error(MAE)

The median absolute error (MAE) is another statistic for measuring dispersion. Like median is more resilient to outliers than the mean, the MAD is also more robust than the standard deviation. MAE is also scale-dependent. And as RMSE, In general, a lower MAE is better than a higher one.

In this study, MAE is calculated by sklearn.metrics.median_absolute_error( ) function.

Results

4.1 Missingness Analysis and Comparison

4.1.1 Missing Rates

The sample size of the original dataset or raw dataset is 552816. Among them, 74560 cases are completed. The ratio of complete cases and all cases is 13.4873%.

The dataset contains 118 variables, 51 are categorical variables. Out of the 51 categorical variables, 43 of them contain missing values. Out of the left 67 numerical variables, 40 of them contains missing values.

All the four simulated datasets, MCAR, MAR, MNAR, and Real contain 74560 cases. MCAR and MAR do not have complete cases, while MNAR has 26858(36.0220%) complete cases; and Real has 20417(27.3833%) complete cases.

The means and standard deviations of each dataset are listed in Table 4.1.

Table 4.1: Missing Rates of Original and Simulated Datasets

	Raw	MCAR	MAR	MNAR	Real
Whole	0.3032	0.3032	0.3050	0.3261	0.2515
	(0.1324)	(0.1323)	(0.1332)	(0.1417)	(0.1096)
Categorical	0.3421	0.3423	0.3440	0.3616	0.2823
	(0.0899)	(0.0899)	(0.0906)	(0.0937)	(0.0728)*
Numerical	0.2615	0.2612	0.2633	0.2878	0.2184
	(0.1572)	(0.1570)	(0.1580)	(0.1728)	(0.1318)

Levene's tests are applied to assess the equality of variances across the five datasets and null hypothesis are accepted, which indicates the variances of groups are equal. Then we performed One-Way ANOVA. Significant between-group differences are found in the missing rates of whole variables ($F = 3.731, p = 0.005$) and with only categorical variables ($F = 5.124, p = 0.001$).

Tukey's HSD test with whole variables rejects the missing rate of Real is no less than that of MNAR at 0.05 level. Tukey's HSD test with only categorical variables indicates the missing rate of Real is less than all other four datasets at 0.05 level. Detailed statistics are attached in the Appendix. These results showed that the simulated Real data obviously has different distribution from other datasets, including the raw dataset. Even if it was generated by the KNN nearest neighbor, and no random missingness spiked in, it's not "Real" at all. But this may be understandable since Real dataset was generated by the nearest neighbor, it by intuition is similar to the complete dataset of the original raw dataset. Thus, the difference indicates that the huge difference between the complete cases and incomplete cases in the EHR data.

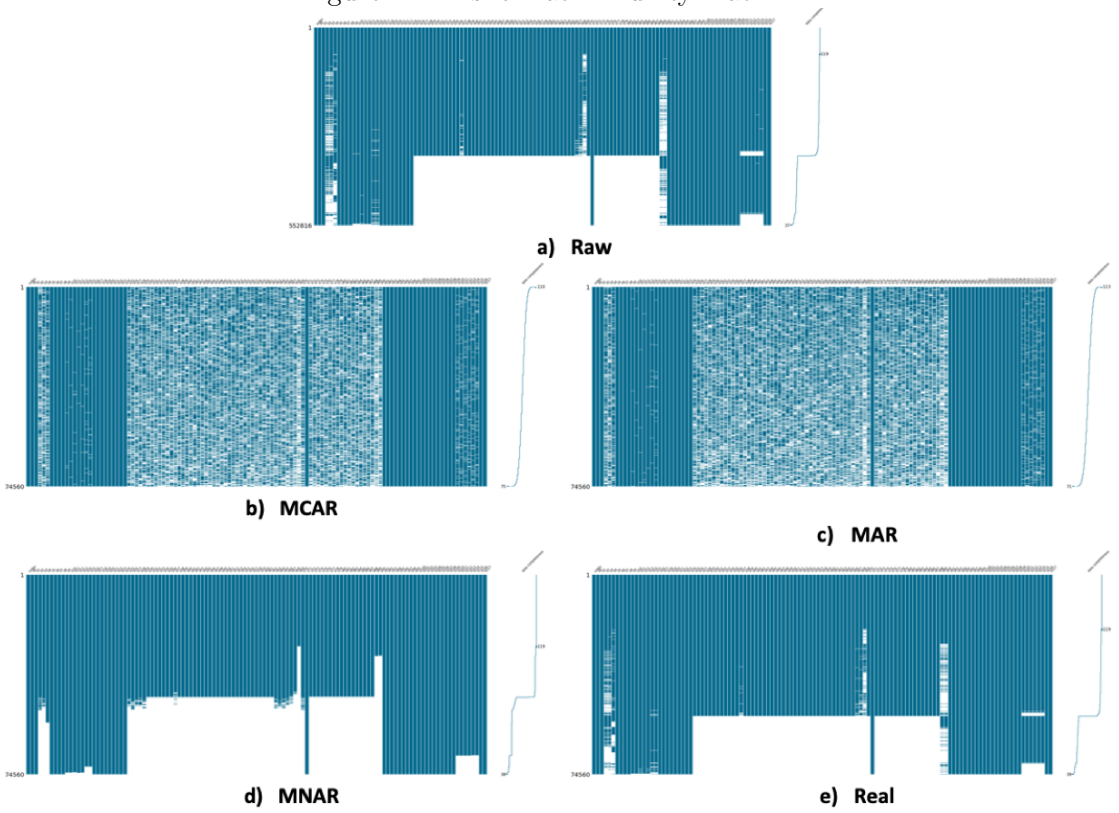
4.1.2 Missingness Distribution Visualization

We also tried different ways to visualize the missingness distribution features.

We used the "missingno" Package in Python, which provides a small toolset of flexible and easy-to-use missing data visualizations and utilities to get a quick visual summary of the completeness.

The msno.matrix nullity matrix is a data-dense display visually presenting patterns in data completion.

Figure 4.1: msno.matrix nullity matrix

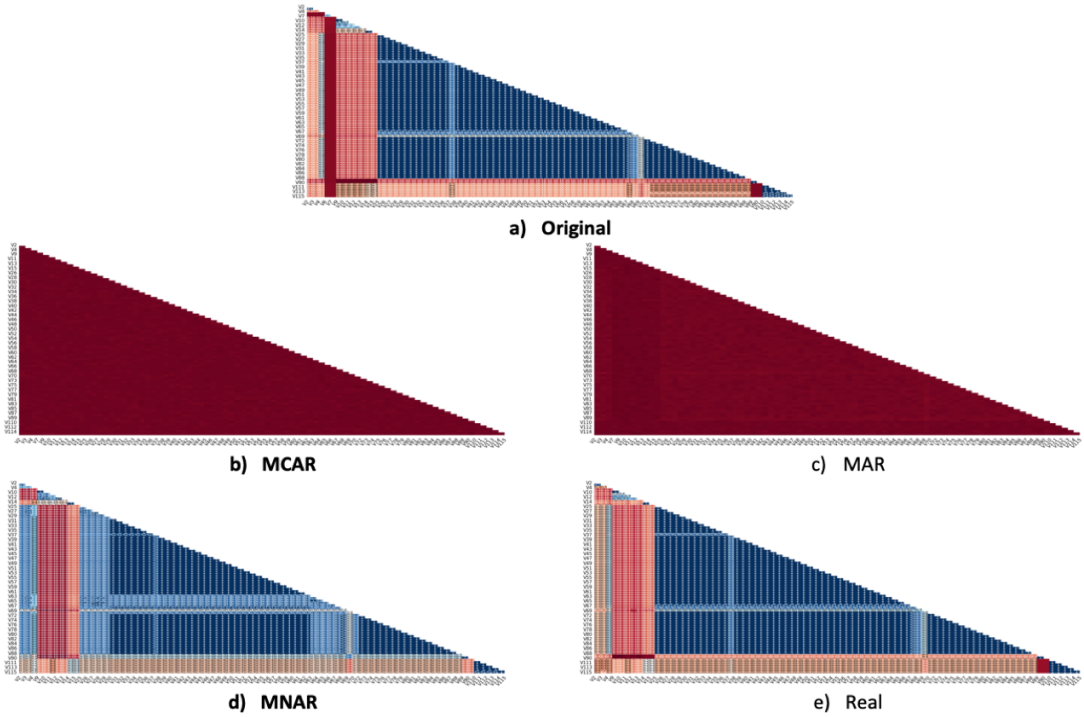


From Figure 4.1, we can see the missingness in MCAR and MAR seems more evenly and randomly, while In the Raw, MNAR and Real dataset, some variables tend to miss together thus form missing chunks.

We also tried the missingno correlation heatmap which measures nullity correlation. In the heatmaps, the denser the color, the stronger the presence or absence of one variable

affects the presence of another. Blue means a positive correlation, and red means a negative correlation.

Figure 4.2: missingno correlation heatmap



Again, we see the heatmaps of MCAR and MAR are nearly even color, which are obviously different from the others.

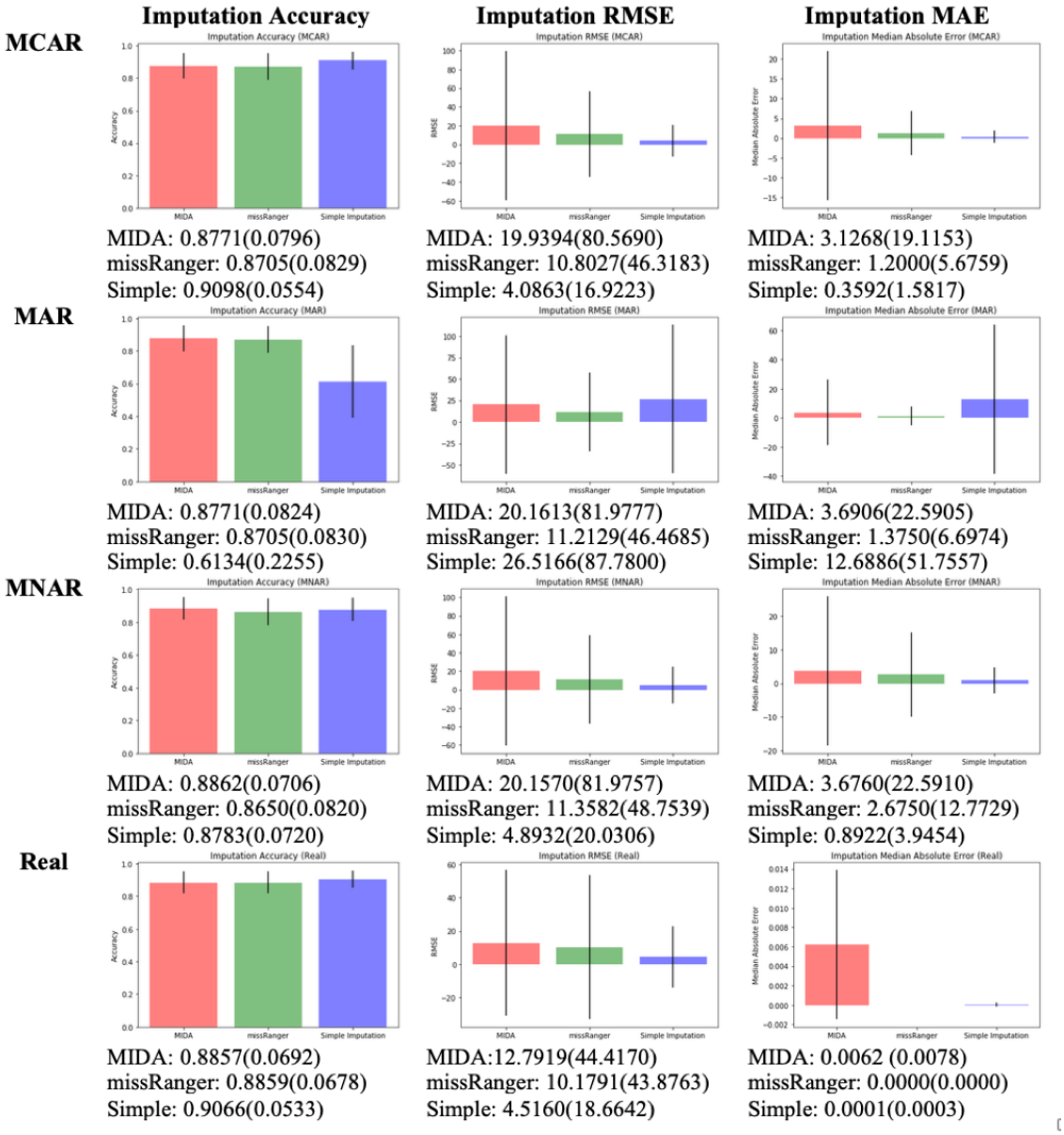
4.2 Imputation Results

4.2.1 Descriptive Results

After Imputation using MIDA, missRanger, and Simple Imputation, we calculated the accuracy scores, F_1 Scores, RMSE and MAE for each imputed dataset. However, the result of F_1 Scores is the same as the accuracy scores, so we only report the accuracy scores in Figure 4.3 below. All the accuracy scores, F_1 Scores, RMSE and MAE by variables in each simulated dataset can be found in the Appendix.

From Figure 4.3, we see our MIDA had done well in categorical variables imputation tasks. It achieved the highest accuracy scores along with smallest standard deviations in both MAR and MNAR datasets. However, it needs improvements for imputing numerical variables. After looking at the documentation of missRanger and Simple Imputation, I think this is at least somehow predictable. As mentioned in missRanger, it avoids imputation with values not already present in the original data. And for simple imputation, no imputed values will jump out of the range of original data too. However, in my MIDA code, at the autoencoder output layer, I intentionally remained a slight probability for getting out-of-ranged values. Another reason for the substandard performance is shown in Figure 2.2. We set the number of training epochs at the point of the first inflection point for PFC. After that point, PFC vibrated, but the RMSEs are decreasing monotonically, which means the cut-point is not optimal for numerical variables.

Figure 4.3: Imputation Result Comparison



4.2.2 One-Way ANOVA and Tukey's HSD Test Results

After performing Levene's tests, One-Way ANOVA test is performed if accepting the null hypothesis of Levene's tests. Otherwise, Kruskal-Wallis H-tests are executed. If any significant

difference is found at .05, we applied the Tukey's HSD Test.

In the MAR dataset, Levene's test result is $L = 29.32, p < .01$. Kruskal-Wallis H-test showed $K = 37.21, p < .01$. Tukey's HSD Test confirmed that the accuracy by simple imputation is lower than both MIDA and missRanger at .05 level.

In the MAR dataset, Levene's test result is $L = 35.32, p < .01$. Kruskal-Wallis H-test showed $K = 54.28, p < .01$. Tukey's HSD Test confirmed that the MAE is greater than both MIDA and missRanger at .05 level.

All test results are attached in the Appendix.

4.2.3 Best Performance Counts

I counted the times that each imputation model beat the other two models. A tie does not count. The results are shown in Table 4.2. Again, we see MIDA did best in the MNAR setting and was a worthy opponent to missRanger in the other three settings. Surprisingly, the simple imputation wins twice: MCAR and Real.

Table 4.2: Best Performance Counts

		MIDA	**missRanger**	**Simple Imputation**
MCAR	Accuracy	3	0	39
	RMSE	1	0	39
	MAE	14	25	1
	Total	18	25	79
MAR	Accuracy	30	12	0
	RMSE	3	37	0
	MAE	14	26	0
	Total	47	75	0
MNAR	Accuracy	28	2	11
	RMSE	6	1	33
	MAE	39	0	1
	Total	73	3	45
Real	Accuracy	3	0	38
	RMSE	0	0	40
	MAE	15	25	0
	Total	18	25	78

Discussion

Our study compared the performance of an autoencoder approach with Random Forest methods and Simple median and mode Imputation on large and realistic dataset. With the large sample size of 74560; the number of variables 118, including 51 categorical variables and 67 numerical variables; the missing rates range from 0.3286% to 63.9780%; four different simulated missingness settings including MCAR, MAR and MNAR, the result may be closer to the real world.

On the contrary of many studies, which experimental results show that all the machine learning methods which we explored outperformed the statistical method (Mean/Mode), based on sensitivity and some cases accuracy[GW17, ALR17]. In our study, simple imputation out stands in the MCAR and Real(KNN) datasets. In the MCAR setting, this is not surprising for mean and mode imputation because missing data is chosen completely at random and is unlikely to have a large effect on statistical averages. And our Real dataset is generate by KNN. As presented in Table 4.1, the missing rates in the Real dataset is significantly lower than other datasets. And by using KNN, it tends to have similar central or typical value for a probability distribution. This may lift the performance of simple Imputation.

In both MAR and MNAR dataset, our MIDA is the best performer. This work is a promising first step in utilizing deep learning techniques for missing data imputation in the EHR, especially when the sample size is 74560, which is at the large end of imputation simulated researches.

Regard to section 4.1.1, which finds out that the "Real" dataset has significant different missing rates than other datasets; and section 4.1.2, where the MCAR and MAR missing distributions are visually different for the Raw, MNAR and Real. We may include that the MNAR dataset is the one that mimic the real data best. In our study, we can say that the neural-base model MIDA beat the tree-based model Random Forest.

However, Challenges still remain.

Firstly, due to time and computational power limits, we only ran the imputation once. More runs of imputation will be executed, thus we will able to calculate the mean and standard deviation of each variable Imputation. When enough data are collected, we will furthermore compare the means and standard deviations as asymptotic bias and variances, which should give a convincing comparison of each model.

Secondly, Autoencoders are computationally intensive, and very flexible. In together, it requires tons of time in hyperparameter tuning. Because we tried to deal with categorical and numerical variables together, the optimal training epochs may differ for the two types of variables. As one of our purposes is to present multiple imputation based on deep denoising autoencoders in Python, more test trials should be done to provide more user-friendly guidelines for hyperparameters choosing.

Last but not least, does complex Imputation really worth it? Non-parametric imputations are computationally expensive and expansive for large datasets, while the simple median/mode Imputation also performed well in most of settings, at least in our study. More studies testing different kinds of datasets will be needed. Obtaining good data quality is crucial for better classification and decision support, and it definitely worth extra effort. However, we can have a relief that even when we don't have enough time or computational power, the simple and elegant median/mode Imputation may not ruin the study!

Appendix

6.1 Missing Rates in different datasets

6.1.1 Missing Rates

Table 6.1 contains the missing rates of each categorical variable in different missing mechanism settings. The total mean and standard deviation of each dataset are reported in Table 4.1.

Table 6.1: Missing Rates of Categorical Variables

	Raw	MCAR	MAR	MNAR	Real
V2	0.346	0.347	0.344	0.346	0.275
V3	0.352	0.352	0.353	0.350	0.279
V4	0.258	0.260	0.257	0.258	0.226
V6	0.000	0.000	0.000	0.000	0.000
V7	0.000	0.000	0.000	0.000	0.000
V25	0.351	0.3525	0.355	0.351	0.291
V26	0.351	0.349	0.353	0.351	0.291
V27	0.351	0.350	0.351	0.351	0.291
V28	0.351	0.353	0.350	0.351	0.291
V29	0.351	0.349	0.352	0.350	0.291

V30	0.351	0.353	0.353	0.386	0.291
V31	0.351	0.347	0.353	0.386	0.291
V32	0.351	0.350	0.353	0.386	0.291
V33	0.351	0.348	0.353	0.386	0.29
V34	0.351	0.350	0.354	0.386	0.291
V35	0.351	0.349	0.353	0.386	0.291
V36	0.351	0.349	0.353	0.386	0.291
V37	0.396	0.397	0.400	0.396	0.321
V63	0.351	0.354	0.352	0.3504	0.291
V64	0.351	0.351	0.353	0.351	0.291
V65	0.351	0.352	0.353	0.350	0.291
V66	0.351	0.351	0.354	0.351	0.291
V67	0.370	0.371	0.367	0.370	0.308
V68	0.399	0.401	0.402	0.396	0.328
V69	0.639	0.639	0.644	0.640	0.513
V70	0.351	0.351	0.353	0.350	0.291
V72	0.352	0.350	0.352	0.387	0.291
V73	0.352	0.353	0.355	0.387	0.291
V74	0.352	0.351	0.353	0.387	0.291
V75	0.352	0.351	0.355	0.387	0.291
V76	0.352	0.354	0.355	0.387	0.291
V77	0.352	0.353	0.355	0.387	0.291
V78	0.352	0.350	0.354	0.387	0.291
V79	0.352	0.351	0.352	0.387	0.291
V80	0.352	0.354	0.354	0.387	0.291
V81	0.352	0.353	0.354	0.387	0.291
V82	0.352	0.353	0.356	0.387	0.291
V83	0.352	0.352	0.353	0.387	0.291

V84	0.352	0.354	0.355	0.387	0.291
V85	0.352	0.355	0.355	0.387	0.291
V86	0.352	0.358	0.356	0.387	0.291
V87	0.352	0.355	0.356	0.387	0.291
V88	0.352	0.351	0.355	0.387	0.290

Table 6.2 contains the missing rates of each numerical variable in different missing mechanism settings. The total mean and standard deviation of each dataset are reported in Table 4.1.

Table 6.2: Missing Rates of Numerical Variables

	Raw	**MCAR**	**MAR**	**MNAR**	**Real**
V9	0.008	0.008	0.008	0.009	0.005
V10	0.008	0.009	0.008	0.009	0.005
V11	0.010	0.011	0.010	0.011	0.007
V12	0.007	0.007	0.007	0.008	0.003
V13	0.009	0.009	0.009	0.008	0.005
V14	0.034	0.035	0.035	0.038	0.034
V15	0.034	0.035	0.036	0.038	0.034
V38	0.351	0.351	0.351	0.386	0.291
V39	0.351	0.348	0.356	0.386	0.291
V40	0.351	0.349	0.352	0.386	0.291
V41	0.351	0.351	0.353	0.386	0.291
V42	0.351	0.351	0.353	0.386	0.291
V43	0.351	0.344	0.352	0.386	0.291
V44	0.351	0.353	0.354	0.386	0.291
V45	0.351	0.349	0.353	0.386	0.291
V46	0.351	0.352	0.351	0.386	0.291
V47	0.351	0.351	0.355	0.386	0.291

V48	0.351	0.352	0.352	0.386	0.291
V49	0.351	0.351	0.353	0.386	0.291
V50	0.351	0.352	0.357	0.386	0.291
V51	0.351	0.351	0.353	0.386	0.291
V52	0.351	0.353	0.355	0.386	0.291
V53	0.351	0.349	0.350	0.386	0.291
V54	0.351	0.349	0.355	0.386	0.291
V55	0.351	0.350	0.352	0.386	0.291
V56	0.351	0.352	0.354	0.386	0.291
V57	0.351	0.350	0.351	0.386	0.291
V58	0.351	0.350	0.351	0.386	0.291
V59	0.351	0.349	0.352	0.386	0.291
V60	0.351	0.352	0.352	0.386	0.291
V61	0.351	0.349	0.353	0.386	0.291
V62	0.351	0.352	0.354	0.386	0.291
V89	0.538	0.536	0.538	0.5925	0.467
V90	0.540	0.537	0.546	0.5945	0.468
V110	0.084	0.083	0.084	0.093	0.073
V111	0.084	0.083	0.084	0.093	0.073
V112	0.084	0.083	0.085	0.093	0.073
V113	0.084	0.083	0.086	0.093	0.073
V114	0.085	0.085	0.086	0.094	0.073
V115	0.086	0.085	0.087	0.095	0.074

Figure 6.1 contains the Tukey HSD test of all the missing rates across four datasets. It is supplementary to the ANOVA results in the section 4.1.1.

Figure 6.1: Tukey's HSD test: missing rate of whole variables

Multiple Comparison of Means - Tukey HSD,FWER=0.05

group1	group2	meandiff	lower	upper	reject
MAR	MCAR	-0.0019	-0.0573	0.0536	False
MAR	MNAR	0.021	-0.0344	0.0764	False
MAR	Raw	-0.0018	-0.0572	0.0536	False
MAR	Real	-0.0536	-0.109	0.0019	False
MCAR	MNAR	0.0229	-0.0326	0.0783	False
MCAR	Raw	0.0	-0.0554	0.0555	False
MCAR	Real	-0.0517	-0.1071	0.0037	False
MNAR	Raw	-0.0228	-0.0782	0.0326	False
MNAR	Real	-0.0745	-0.13	-0.0191	True
Raw	Real	-0.0517	-0.1071	0.0037	False

Figure 6.2 contains the Tukey HSD test of all the missing rates of categorical variables across four datasets. It is supplementary to the ANOVA results in the section 4.1.1.

Figure 6.2: Tukey's HSD test: missing rate of categorical variables

Multiple Comparison of Means - Tukey HSD,FWER=0.05

group1	group2	meandiff	lower	upper	reject
MAR	MCAR	-0.0017	-0.0537	0.0504	False
MAR	MNAR	0.0176	-0.0344	0.0697	False
MAR	Raw	-0.0019	-0.054	0.0501	False
MAR	Real	-0.0617	-0.1137	-0.0096	True
MCAR	MNAR	0.0193	-0.0328	0.0713	False
MCAR	Raw	-0.0003	-0.0523	0.0518	False
MCAR	Real	-0.06	-0.1121	-0.008	True
MNAR	Raw	-0.0195	-0.0716	0.0325	False
MNAR	Real	-0.0793	-0.1313	-0.0273	True
Raw	Real	-0.0598	-0.1118	-0.0077	True

6.2 MCAR

Table 6.3 contains the accuracy scores and F_1 scores of each categorical variable using MIDA, missRanger and simple imputation in MCAR. The total mean and standard deviation of accuracy scores of each imputation methods are list in Figure 4.3.

6.2.1 MCAR: Categorical Variables Results

Table 6.3: MCAR - Categorical Variables Results

	Accuracy			**F1 Scores**		
	MIDA	missRanger	Simple	MIDA	missRanger	Simple
V2	0.8410	0.8418	0.8613	0.8410	0.8418	0.8613
V3	0.7832	0.7710	0.8288	0.7832	0.7710	0.8288
V4	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
V7	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
V25	0.7604	0.7433	0.9264	0.7604	0.7433	0.9264
V26	0.8666	0.8411	0.9481	0.8666	0.8411	0.9481
V27	0.9800	0.9783	0.9796	0.9800	0.9783	0.9796
V28	0.9438	0.9414	0.9496	0.9438	0.9414	0.9496
V29	0.7857	0.7804	0.8445	0.7857	0.7804	0.8445
V30	0.8415	0.8415	0.8516	0.8415	0.8415	0.8516
V31	0.9941	0.9941	0.9943	0.9941	0.9941	0.9943
V32	0.7583	0.7602	0.8224	0.7583	0.7602	0.8224
V33	0.9346	0.9346	0.9353	0.9346	0.9346	0.9353
V34	0.7696	0.7723	0.8291	0.7696	0.7723	0.8291
V35	0.9269	0.9269	0.9428	0.9269	0.9269	0.9428
V36	0.8906	0.8906	0.8990	0.8906	0.8906	0.8990
V37	0.7716	0.7249	0.8510	0.7716	0.7249	0.8510

Table 6.3: MCAR - Categorical Variables Results

	Accuracy			F1 Scores		
V63	0.9190	0.8080	0.9500	0.9190	0.8080	0.9500
V64	0.8807	0.8866	0.9012	0.8807	0.8866	0.9012
V65	0.9004	0.8092	0.9031	0.9004	0.8092	0.9031
V66	0.9918	0.9918	0.9920	0.9918	0.9918	0.9920
V67	0.8798	0.8983	0.9066	0.8798	0.8983	0.9066
V68	0.9388	0.9388	0.9452	0.9388	0.9388	0.9452
V69	0.7407	0.7604	0.7833	0.7407	0.7604	0.7833
V70	0.8218	0.8049	0.8557	0.8218	0.8049	0.8557
V72	0.8784	0.8788	0.8871	0.8784	0.8788	0.8871
V73	0.7859	0.7859	0.8797	0.7859	0.7859	0.8797
V74	0.8599	0.8599	0.8836	0.8599	0.8599	0.8836
V75	0.7862	0.8253	0.8415	0.7862	0.8253	0.8415
V76	0.9009	0.9009	0.9069	0.9009	0.9009	0.9069
V77	0.7537	0.7610	0.8753	0.7537	0.7610	0.8753
V78	0.9446	0.9446	0.9463	0.9446	0.9446	0.9463
V79	0.9492	0.9492	0.9513	0.9492	0.9492	0.9513
V80	0.8888	0.8900	0.9140	0.8888	0.8900	0.9140
V81	0.8142	0.7733	0.9010	0.8142	0.7733	0.9010
V82	0.9006	0.9006	0.9320	0.9006	0.9006	0.9320
V83	0.9708	0.9708	0.9754	0.9708	0.9708	0.9754
V84	0.8042	0.8042	0.8756	0.8042	0.8042	0.8756
V85	0.8401	0.8401	0.8845	0.8401	0.8401	0.8845
V86	0.9618	0.9618	0.9719	0.9618	0.9618	0.9719
V87	0.8795	0.8795	0.8895	0.8795	0.8795	0.8895
V88	0.9964	0.9964	0.9965	0.9964	0.9964	0.9965

6.2.2 MCAR: Numerical Variables Results

Table 6.4 contains the RMSE and MAE of each numerical variable using MIDA, missRanger and simple imputation in MCAR. The total mean and standard deviation of RMSE and MAE of each imputation methods are list in Figure 4.3.

Table 6.4: MCAR - Numerical Variables Results

	Root Mean Square Deviation			**Median Absolute Error**		
	MIDA	missRanger	Simple	MIDA	missRanger	Simple
V9	11.4809	1.8631	1.5097	0.000	0.0	0.000
V10	6.6354	1.1891	0.8506	0.000	0.0	0.000
V11	0.2323	0.2316	0.2311	0.000	0.0	0.000
V12	6.7251	1.2535	1.2130	0.000	0.0	0.000
V13	0.1833	0.1947	0.1842	0.000	0.0	0.000
V14	67.7952	1.6586	1.5743	0.000	0.0	0.000
V15	0.3249	0.0630	0.0313	0.000	0.0	0.000
V38	0.2431	0.2429	0.2137	0.0047	0.0	0.0076
V39	0.3779	0.3719	0.3141	0.0248	0.0	0.0047
V40	0.1434	0.1415	0.0895	0.0051	0.0	0.0006
V41	0.1554	0.1537	0.1376	0.0086	0.0	0.000
V42	0.1548	0.1527	0.1220	0.0098	0.0	0.000
V43	0.0920	0.0909	0.0824	0.0040	0.0	0.0031
V44	0.0986	0.0973	0.0816	0.0053	0.0	0.000
V45	0.1662	0.1641	0.1539	0.0101	0.0	0.000
V46	0.0715	0.0708	0.0383	0.0027	0.0	0.000
V47	0.1336	0.1319	0.0933	0.0052	0.0	0.0035
V48	0.0976	0.0960	0.0574	0.0101	0.0	0.000
V49	0.1460	0.1466	0.1039	0.0031	0.0	0.0008
V50	0.2337	0.2370	0.1759	0.0051	0.0	0.0058

Table 6.4: MCAR - Numerical Variables Results

	Root Mean Square Deviation			**Median Absolute Error**		
V51	0.1835	0.1885	0.1264	0.0107	0.0	0.0014
V52	0.1250	0.1244	0.0918	0.0012	0.0	0.0032
V53	0.1434	0.1415	0.1138	0.0061	0.0	0.0011
V54	0.1819	0.1831	0.1279	0.0016	0.0	0.0012
V55	0.0105	0.0105	0.0101	0.0001	0.0	0.000
V56	0.2327	0.2335	0.1683	0.0022	0.0	0.0037
V57	0.1921	0.1895	0.1419	0.0115	0.0	0.000
V58	0.2030	0.2026	0.1530	0.0031	0.0	0.0013
V59	0.0575	0.0569	0.0468	0.0030	0.0	0.000
V60	0.0467	0.0457	0.0419	0.0070	0.0	0.000
V61	0.1244	0.1229	0.0917	0.0066	0.0	0.000
V62	0.0811	0.0800	0.0712	0.0068	0.0	0.000
V89	212.5236	182.1649	71.9638	3.9794	15.0	7.2915
V90	467.9908	234.8182	81.5442	120.9368	33.0	7.0404
V110	3.2446	0.9445	0.3053	0.000	0.0	0.000
V111	3.5824	0.7305	0.2374	0.000	0.0	0.000
V112	3.0509	0.7870	0.1020	0.000	0.0	0.000
V113	3.4535	0.8144	0.1990	0.000	0.0	0.000
V114	3.3956	0.9235	0.3804	0.000	0.0	0.000
V115	3.2629	0.7940	0.2784	0.000	0.0	0.000

6.3 MAR

Table 6.5 contains the accuracy scores and F_1 scores of each categorical variable using MIDA, missRanger and simple imputation in MAR. The total mean and standard deviation of accuracy scores of each imputation methods are list in Figure 4.3.

6.3.1 MAR: Categorical Variables Results

Table 6.5: MAR - Categorical Variables Results

	Accuracy			**F1 Scores**		
	MIDA	missRanger	Simple	MIDA	missRanger	Simple
V2	0.8407	0.8416	0.4704	0.8407	0.8416	0.4704
V3	0.7728	0.7697	0.2366	0.7728	0.7697	0.2366
V4	1.0000	1.0000	0.9999	1.0000	1.0000	0.9999
V7	1.0000	1.0000	0.9394	1.0000	1.0000	0.9394
V25	0.8547	0.7438	0.1547	0.8547	0.7438	0.1547
V26	0.8569	0.8412	0.5081	0.8569	0.8412	0.5081
V27	0.9767	0.9776	0.8971	0.9767	0.9776	0.8971
V28	0.9427	0.9427	0.7444	0.9427	0.9427	0.7444
V29	0.7647	0.7788	0.2830	0.7647	0.7788	0.2830
V30	0.8439	0.8439	0.5385	0.8439	0.8439	0.5385
V31	0.9942	0.9942	0.9831	0.9942	0.9942	0.9831
V32	0.7249	0.7586	0.4669	0.7249	0.7586	0.4669
V33	0.9359	0.9359	0.7908	0.9359	0.9359	0.7908
V34	0.7873	0.7731	0.4537	0.7873	0.7731	0.4537
V35	0.9250	0.9250	0.7528	0.9250	0.9250	0.7528
V36	0.8872	0.8924	0.6530	0.8872	0.8924	0.6530
V37	0.7364	0.7230	0.2531	0.7364	0.7230	0.2531

Table 6.5: MAR - Categorical Variables Results

	Accuracy			F1 Scores		
V63	0.9188	0.8082	0.3727	0.9188	0.8082	0.3727
V64	0.8870	0.8873	0.5572	0.8870	0.8873	0.5572
V65	0.9102	0.8096	0.3576	0.9102	0.8096	0.3576
V66	0.9917	0.9917	0.9636	0.9917	0.9917	0.9636
V67	0.8962	0.8962	0.6365	0.8962	0.8962	0.6365
V68	0.9401	0.9401	0.7733	0.9401	0.9401	0.7733
V69	0.7515	0.7602	0.5370	0.7515	0.7602	0.5370
V70	0.8036	0.8044	0.3020	0.8036	0.8044	0.3020
V72	0.8756	0.8789	0.6254	0.8756	0.8789	0.6254
V73	0.7898	0.7898	0.4659	0.7898	0.7898	0.4659
V74	0.8601	0.8601	0.5722	0.8601	0.8601	0.5722
V75	0.8149	0.8280	0.5103	0.8149	0.8280	0.5103
V76	0.9031	0.9031	0.6877	0.9031	0.9031	0.6877
V77	0.6977	0.7573	0.4607	0.6977	0.7573	0.4607
V78	0.9402	0.9402	0.8057	0.9402	0.9402	0.8057
V79	0.9495	0.9495	0.8284	0.9495	0.9495	0.8284
V80	0.8980	0.8918	0.6567	0.8980	0.8918	0.6567
V81	0.8112	0.7715	0.4595	0.8112	0.7715	0.4595
V82	0.8976	0.8978	0.6771	0.8976	0.8978	0.6771
V83	0.9710	0.9710	0.9006	0.9710	0.9710	0.9006
V84	0.8126	0.8083	0.4782	0.8126	0.8083	0.4782
V85	0.8380	0.8380	0.5320	0.8380	0.8380	0.5320
V86	0.9632	0.9632	0.8705	0.9632	0.9632	0.8705
V87	0.8764	0.8764	0.6206	0.8764	0.8764	0.6206
V88	0.9964	0.9964	0.9869	0.9964	0.9964	0.9869

6.3.2 MAR: Numerical Variables Results

Table 6.6 contains the RMSE and MAE of each numerical variable using MIDA, missRanger and simple imputation in MAR. The total mean and standard deviation of RMSE and MAE of each imputation methods are list in Figure 4.3.

Table 6.6: MAR - Numerical Variables Results

	Root Mean Square Deviation			**Median Absolute Error**		
	MIDA	missRanger	Simple	MIDA	missRanger	Simple
V9	12.0581	1.8328	27.9587	0.000	0.0	19.0000
V10	6.5791	1.0994	17.4315	0.000	0.0	12.0000
V11	0.1340	0.0694	2.4833	0.000	0.0	0.6000
V12	6.9001	1.2432	20.4903	0.000	0.0	13.0000
V13	1.4975	0.1897	3.3382	0.000	0.0	2.0000
V14	70.7070	16.9142	225.4968	0.000	0.0	6.7200
V15	0.3273	0.0635	0.4693	0.000	0.0	0.3000
V38	0.2428	0.2422	0.4274	0.0036	0.0	0.0076
V39	0.3845	0.3784	0.6833	0.0248	0.0	0.000
V40	0.1417	0.1402	0.2414	0.0041	0.0	0.1583
V41	0.1556	0.1539	0.2850	0.0086	0.0	0.000
V42	0.1585	0.1564	0.2846	0.0098	0.0	0.000
V43	0.0922	0.0911	0.1636	0.0040	0.0	0.000
V44	0.1003	0.0991	0.1789	0.0053	0.0	0.000
V45	0.1674	0.1653	0.3024	0.0101	0.0	0.000
V46	0.0726	0.0718	0.1298	0.0027	0.0	0.000
V47	0.1324	0.1323	0.2351	0.0013	0.0	0.0035
V48	0.0977	0.0961	0.1850	0.0101	0.0	0.000
V49	0.1492	0.1454	0.2682	0.0123	0.0	0.000
V50	0.2332	0.2364	0.3984	0.0050	0.0	0.1756

Table 6.6: MAR - Numerical Variables Results

	Root Mean Square Deviation			**Median Absolute Error**		
V51	0.1898	0.1859	0.2984	0.0066	0.0	0.2598
V52	0.1240	0.1241	0.2091	0.0008	0.0	0.0528
V53	0.1416	0.1411	0.2537	0.0021	0.0	0.000
V54	0.1817	0.1828	0.3015	0.0016	0.0	0.2950
V55	0.0096	0.0092	0.0179	0.0025	0.0	0.000
V56	0.2413	0.2404	0.4015	0.0020	0.0	0.2998
V57	0.1902	0.1875	0.3504	0.0115	0.0	0.000
V58	0.2136	0.2071	0.3862	0.0249	0.0	0.000
V59	0.0590	0.0583	0.1070	0.0030	0.0	0.000
V60	0.0452	0.0442	0.0849	0.0070	0.0	0.000
V61	0.1277	0.1262	0.2332	0.0066	0.0	0.000
V62	0.0798	0.0787	0.1434	0.0068	0.0	0.000
V89	205.8112	182.4331	304.3561	4.5224	15.0	134.2537
V90	480.3093	235.9316	428.4068	142.9243	40.0	303.5153
V110	3.3509	0.9494	4.4688	0.000	0.0	3.0000
V111	2.386	0.7320	3.4520	0.000	0.0	2.0000
V112	2.7656	0.7929	3.7659	0.000	0.0	2.4019
V113	2.5609	0.8358	3.8750	0.000	0.0	2.5000
V114	3.5991	0.9289	4.2608	0.000	0.0	3.0000
V115	3.7331	0.8069	3.8406	0.000	0.0	2.0000

6.4 MNAR

Table 6.7 contains the accuracy scores and F_1 scores of each categorical variable using MIDA, missRanger and simple imputation in MNAR. The total mean and standard deviation of accuracy scores of each imputation methods are list in Figure 4.3.

6.4.1 MNAR: Categorical Variables Results

Table 6.7: MNAR - Categorical Variables Results

	Accuracy			**F1 Scores**		
	MIDA	missRanger	Simple	MIDA	missRanger	Simple
V2	0.8407	0.8271	0.8550	0.8407	0.8271	0.8550
V3	0.7728	0.7833	0.7909	0.7728	0.7833	0.7909
V4	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
V25	0.8547	0.7565	0.9217	0.8547	0.7565	0.9217
V26	0.8569	0.8241	0.8616	0.8569	0.8241	0.8616
V27	0.9767	0.9758	0.9779	0.9767	0.9758	0.9779
V28	0.9427	0.9356	0.9375	0.9427	0.9356	0.9375
V29	0.7647	0.7941	0.7903	0.7647	0.7941	0.7903
V30	0.8514	0.8371	0.8485	0.8514	0.8371	0.8485
V31	0.9979	0.9935	0.9937	0.9979	0.9935	0.9937
V32	0.8380	0.7444	0.8077	0.8380	0.7444	0.8077
V33	0.9405	0.9358	0.9366	0.9405	0.9358	0.9366
V34	0.8036	0.7602	0.8098	0.8036	0.7602	0.8098
V35	0.9301	0.9240	0.9312	0.9301	0.9240	0.9312
V36	0.8996	0.8856	0.8953	0.8996	0.8856	0.8953
V37	0.7364	0.7375	0.7781	0.7364	0.7375	0.7781
V63	0.9188	0.8259	0.7862	0.9188	0.8259	0.7862

Table 6.7: MNAR - Categorical Variables Results

	Accuracy			F1 Scores		
V64	0.8870	0.8747	0.8791	0.8870	0.8747	0.8791
V65	0.9102	0.8253	0.7869	0.9102	0.8253	0.7869
V66	0.9917	0.9913	0.9915	0.9917	0.9913	0.9915
V67	0.8962	0.8876	0.8891	0.8962	0.8876	0.8891
V68	0.9401	0.9339	0.9390	0.9401	0.9339	0.9390
V69	0.7515	0.7371	0.7584	0.7515	0.7371	0.7584
V70	0.8036	0.8214	0.7945	0.8036	0.8214	0.7945
V72	0.8862	0.8788	0.8876	0.8862	0.8788	0.8876
V73	0.7975	0.7803	0.8045	0.7975	0.7803	0.8045
V74	0.8677	0.8536	0.8596	0.8677	0.8536	0.8596
V75	0.8367	0.8227	0.8293	0.8367	0.8227	0.8293
V76	0.9094	0.9046	0.9081	0.9094	0.9046	0.9081
V77	0.8781	0.7442	0.8356	0.8781	0.7442	0.8356
V78	0.9447	0.9423	0.9441	0.9447	0.9423	0.9441
V79	0.9535	0.9493	0.9512	0.9535	0.9493	0.9512
V80	0.9101	0.8901	0.8955	0.9101	0.8901	0.8955
V81	0.8495	0.7561	0.8361	0.8495	0.7561	0.8361
V82	0.9044	0.8985	0.9040	0.9044	0.8985	0.9040
V83	0.9733	0.9718	0.9727	0.9733	0.9718	0.9727
V84	0.8275	0.7951	0.8223	0.8275	0.7951	0.8223
V85	0.8462	0.8348	0.7842	0.8462	0.8348	0.7842
V86	0.9659	0.9631	0.9637	0.9659	0.9631	0.9637
V87	0.8832	0.8737	0.8557	0.8832	0.8737	0.8557
V88	0.9967	0.9963	0.9963	0.9967	0.9963	0.9963

6.4.2 MNAR: Numerical Variables Results

Table 6.8 contains the RMSE and MAE of each numerical variable using MIDA, missRanger and simple imputation in MNAR. The total mean and standard deviation of RMSE and MAE of each imputation methods are list in Figure 4.3.

Table 6.8: MNAR - Numerical Variables Results

	Root Mean Square Deviation			Median Absolute Error		
	MIDA	missRanger	Simple	MIDA	missRanger	Simple
V9	12.0581	1.8670	1.8149	0.000	0.0	0.000
V10	6.5791	1.2337	1.1711	0.000	0.0	0.000
V11	0.1340	0.0705	0.0710	0.000	0.0	0.000
V12	6.9001	1.4173	1.3316	0.000	0.0	0.000
V13	1.4975	0.1862	0.1860	0.000	0.0	0.000
V14	70.7070	1.7269	1.5000	0.000	0.0	0.000
V15	0.3273	0.0652	0.0568	0.000	0.0	0.000
V38	0.2347	0.2445	0.2294	0.000	0.0	0.0012
V39	0.3718	0.3763	0.3398	0.000	0.0	0.0021
V40	0.1382	0.1435	0.1056	0.000	0.0	0.000
V41	0.1494	0.1565	0.1484	0.000	0.0	0.000
V42	0.1521	0.1545	0.1483	0.000	0.0	0.000
V43	0.0888	0.0909	0.0849	0.000	0.0	0.0031
V44	0.0964	0.0966	0.0913	0.000	0.0	0.000
V45	0.1604	0.1628	0.1579	0.000	0.0	0.000
V46	0.0698	0.0720	0.0682	0.000	0.0	0.000
V47	0.1284	0.1325	0.1231	0.000	0.0	0.0028
V48	0.0932	0.0986	0.0935	0.000	0.0	0.000
V49	0.1443	0.1493	0.1363	0.000	0.0	0.000
V50	0.2279	0.2396	0.2044	0.000	0.0	0.0023

Table 6.8: MNAR - Numerical Variables Results

	Root Mean Square Deviation			Median Absolute Error		
V51	0.1863	0.1903	0.1446	0.000	0.0	0.000
V52	0.1213	0.1266	0.1119	0.000	0.0	0.0015
V53	0.1366	0.1430	0.1337	0.000	0.0	0.000
V54	0.1780	0.1865	0.1541	0.000	0.0	0.0013
V55	0.0090	0.0093	0.0092	0.000	0.0	0.000
V56	0.2359	0.2373	0.2077	0.000	0.0	0.0018
V57	0.1825	0.1906	0.1846	0.000	0.0	0.000
V58	0.2051	0.2142	0.2063	0.000	0.0	0.0008
V59	0.0566	0.0589	0.0569	0.000	0.0	0.000
V60	0.0429	0.0440	0.0415	0.000	0.0	0.000
V61	0.1225	0.1246	0.1207	0.000	0.0	0.000
V62	0.0764	0.0779	0.0764	0.000	0.0	0.000
V89	205.7638	191.1763	85.8278	4.114	32.0	16.694
V90	480.3093	247.5827	96.0554	142.924	75.0	18.980
V110	3.3509	1.0043	0.7987	0.000	0.0	0.000
V111	2.386	0.7676	0.6459	0.000	0.0	0.000
V112	2.7656	0.8321	0.6736	0.000	0.0	0.000
V113	2.5609	0.8653	0.7071	0.000	0.0	0.000
V114	3.5991	0.9716	0.8064	0.000	0.0	0.000
V115	3.7331	0.8401	0.7047	0.000	0.0	

6.5 Real

Table 6.9 contains the accuracy scores and F_1 scores of each categorical variable using MIDA, missRanger and simple imputation in Real. The total mean and standard deviation of accuracy scores of each imputation methods are list in Figure 4.3.

6.5.1 Real: Categorical Variables Results

Table 6.9: Real - Categorical Variables Results

	Accuracy			**F1 Scores**		
V	MIDA	missRanger	Simple	MIDA	missRanger	Simple
V2	0.8798	0.8839	0.8931	0.8798	0.8839	0.8931
V3	0.7486	0.8113	0.8616	0.7486	0.8113	0.8616
V4	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
V25	0.8772	0.7918	0.9400	0.8772	0.7918	0.9400
V26	0.8710	0.8710	0.9569	0.8710	0.8710	0.9569
V27	0.9814	0.9814	0.9834	0.9814	0.9814	0.9834
V28	0.9520	0.9520	0.9552	0.9520	0.9520	0.9552
V29	0.8159	0.8176	0.8712	0.8159	0.8176	0.8712
V30	0.8553	0.8553	0.8657	0.8553	0.8553	0.8657
V31	0.9947	0.9947	0.9950	0.9947	0.9947	0.9950
V32	0.7937	0.7863	0.8467	0.7937	0.7863	0.8467
V33	0.9393	0.9393	0.9397	0.9393	0.9393	0.9397
V34	0.7945	0.7983	0.8420	0.7945	0.7983	0.8420
V35	0.9300	0.9300	0.9370	0.9300	0.9300	0.9370
V36	0.8954	0.8955	0.9051	0.8954	0.8955	0.9051
V37	0.7816	0.7779	0.8508	0.7816	0.7779	0.8508
V63	0.8396	0.8435	0.8771	0.8396	0.8435	0.8771

Table 6.9: Real - Categorical Variables Results

	Accuracy			F1 Scores		
V64	0.9052	0.9054	0.9183	0.9052	0.9054	0.9183
V65	0.8405	0.8430	0.8740	0.8405	0.8430	0.8740
V66	0.9934	0.9934	0.9935	0.9934	0.9934	0.9935
V67	0.9123	0.9151	0.9182	0.9123	0.9151	0.9182
V68	0.9540	0.9540	0.9570	0.9540	0.9540	0.9570
V69	0.8100	0.8116	0.8350	0.8100	0.8116	0.8350
V70	0.8277	0.8390	0.8734	0.8277	0.8390	0.8734
V72	0.8946	0.8954	0.9028	0.8946	0.8954	0.9028
V73	0.8175	0.8175	0.8342	0.8175	0.8175	0.8342
V74	0.8740	0.8740	0.8791	0.8740	0.8740	0.8791
V75	0.8434	0.8434	0.8486	0.8434	0.8434	0.8486
V76	0.9046	0.9046	0.9080	0.9046	0.9046	0.9080
V77	0.8166	0.7949	0.8633	0.8166	0.7949	0.8633
V78	0.9439	0.9439	0.9456	0.9439	0.9439	0.9456
V79	0.9527	0.9527	0.9542	0.9527	0.9527	0.9542
V80	0.9037	0.9037	0.9096	0.9037	0.9037	0.9096
V81	0.7735	0.8043	0.8695	0.7735	0.8043	0.8695
V82	0.9049	0.9049	0.9093	0.9049	0.9049	0.9093
V83	0.9739	0.9739	0.9745	0.9739	0.9739	0.9745
V84	0.8217	0.8228	0.8449	0.8217	0.8228	0.8449
V85	0.8497	0.8497	0.8043	0.8497	0.8497	0.8043
V86	0.9669	0.9669	0.9677	0.9669	0.9669	0.9677
V87	0.8822	0.8822	0.8684	0.8822	0.8822	0.8684
V88	0.9966	0.9966	0.9967	0.9966	0.9966	0.9967

6.5.2 Real: Numerical Variables Results

Table 6.10 contains the RMSE and MAE of each numerical variable using MIDA, missRanger and simple imputation in Real. The total mean and standard deviation of RMSE and MAE of each imputation methods are list in Figure 4.3.

Table 6.10: Real-Numerical Variables Results

	Root Mean Square Deviation			**Median Absolute Error**		
	MIDA	missRanger	Simple	MIDA	missRanger	Simple
V9	7.8371	1.3334	1.2239	0.000	0.0	0.000
V10	4.1763	0.8390	0.7537	0.000	0.0	0.000
V11	6.5008	0.0543	0.0538	0.000	0.0	0.000
V12	4.2032	0.8291	0.7688	0.000	0.0	0.000
V13	1.1226	0.1447	0.1394	0.000	0.0	0.000
V14	66.8290	1.6480	1.6136	0.000	0.0	0.000
V15	0.3120	0.0614	0.0523	0.000	0.0	0.000
V38	0.2257	0.2237	0.2018	0.0094	0.0	0.000
V39	0.3670	0.3611	0.3285	0.0248	0.0	0.0010
V40	0.1254	0.1315	0.0953	0.0179	0.0	0.000
V41	0.1486	0.1469	0.1395	0.0086	0.0	0.000
V42	0.1445	0.1425	0.1330	0.0098	0.0	0.000
V43	0.0903	0.0892	0.0837	0.0040	0.0	0.0012
V44	0.0955	0.0942	0.0892	0.0053	0.0	0.000
V45	0.1655	0.1635	0.1563	0.0101	0.0	0.000
V46	0.0683	0.0675	0.0646	0.0027	0.0	0.000
V47	0.1250	0.1241	0.1167	0.0052	0.0	0.000
V48	0.0928	0.0913	0.0854	0.0101	0.0	0.000
V49	0.1418	0.1383	0.1248	0.0123	0.0	0.000
V50	0.2084	0.2092	0.1785	0.0077	0.0	0.000

V51	0.1572	0.1683	0.1247	0.0101	0.0	0.000
V52	0.1133	0.1158	0.1015	0.0030	0.0	0.000
V53	0.1361	0.1342	0.1259	0.0061	0.0	0.000
V54	0.1519	0.1657	0.1379	0.0337	0.0	0.000
V55	0.0111	0.0107	0.0106	0.0025	0.0	0.000
V56	0.2149	0.2174	0.1904	0.0049	0.0	0.000
V57	0.1793	0.1768	0.1703	0.0115	0.0	0.000
V58	0.1991	0.1928	0.1831	0.0249	0.0	0.000
V59	0.0575	0.0569	0.0547	0.0030	0.0	0.000
V60	0.0457	0.0447	0.0429	0.0070	0.0	0.000
V61	0.1233	0.1219	0.1176	0.0066	0.0	0.000
V62	0.0746	0.0736	0.0718	0.0068	0.0	0.000
V89	197.4789	170.3589	77.7888	0.000	0.0	0.000
V90	199.5349	224.0377	91.3362	0.000	0.0	0.000
V110	3.1510	0.8177	0.7019	0.000	0.0	0.000
V111	3.5521	0.6563	0.5613	0.000	0.0	0.000
V112	3.3522	0.7002	0.5955	0.000	0.0	0.000
V113	3.3690	0.7221	0.6193	0.000	0.0	0.000
V114	3.3496	0.7903	0.6862	0.000	0.0	0.000
V115	3.4440	0.7108	0.6162	0.000	0.0	0.000

6.6 One-Way ANOVA and Tukey's HSD Test Results

Figure 6.3 contains the results of Levenes test, ANOVA test(if null hypothesis of Levenes test accepted), Kruskal-Wallis H-test(if null hypothesis of Levenes test rejected), and Tukey HSD test(if ANOVA test or Kruskal-Wallis H-test found significant differences) of MCAR. It is supplementary to the section 4.2.2.

6.6.1 MCAR: One-Way ANOVA and Tukey's HSD Test Results

Figure 6.3: MCAR: One-Way ANOVA and Tukey's HSD Test Results

```
accuracy
Levene test
LeveneResult(statistic=5.109440677710103, pvalue=0.007385309953146567)
F_onewayResult
F_onewayResult(statistic=3.4290746005370067, pvalue=0.03554741394323934)
KruskalResult
KruskalResult(statistic=6.1983969133929655, pvalue=0.04508532575535051)
         Multiple Comparison of Means - Tukey HSD,FWER=0.05
=============================================================================
     group1             group2        meandiff  lower  upper  reject
-----------------------------------------------------------------------------
MCAR_da_accuracy   MCAR_rf_accuracy   -0.0065  -0.0447 0.0316 False
MCAR_da_accuracy MCAR_simple_accuracy  0.0328  -0.0054 0.0709 False
MCAR_rf_accuracy MCAR_simple_accuracy  0.0393   0.0011 0.0774  True
-----------------------------------------------------------------------------

f1_score
Levene test
LeveneResult(statistic=5.109440677710101, pvalue=0.007385309953146567)
F_onewayResult
F_onewayResult(statistic=3.4290746005370054, pvalue=0.03554741394323934)
KruskalResult
         Multiple Comparison of Means - Tukey HSD,FWER=0.05
=============================================================================
     group1             group2        meandiff  lower  upper  reject
-----------------------------------------------------------------------------
MCAR_da_f1_score   MCAR_rf_f1_score   -0.0065  -0.0447 0.0316 False
MCAR_da_f1_score MCAR_simple_f1_score  0.0328  -0.0054 0.0709 False
MCAR_rf_f1_score MCAR_simple_f1_score  0.0393   0.0011 0.0774  True
-----------------------------------------------------------------------------

rmse
Levene test
LeveneResult(statistic=0.8481944469005154, pvalue=0.43080305706234845)
F_onewayResult
F_onewayResult(statistic=0.8515192453460841, pvalue=0.4293935448225874)
KruskalResult
KruskalResult(statistic=4.937148760330558, pvalue=0.08470553084992696)
      Multiple Comparison of Means - Tukey HSD,FWER=0.05
=================================================================
   group1         group2      meandiff  lower    upper  reject
-----------------------------------------------------------------
MCAR_da_rmse   MCAR_rf_rmse   -9.1367  -38.0888 19.8153 False
MCAR_da_rmse MCAR_simple_rmse -15.8531 -44.8051  13.099 False
MCAR_rf_rmse MCAR_simple_rmse -6.7163  -35.6684 22.2357 False
-----------------------------------------------------------------

median_absolute_error
Levene test
LeveneResult(statistic=0.6034974723314075, pvalue=0.5485889637717415)
F_onewayResult
F_onewayResult(statistic=0.6038026848507849, pvalue=0.548423262677541)
KruskalResult
KruskalResult(statistic=31.471450798305646, pvalue=1.4657546829956366e-07)
                     Multiple Comparison of Means - Tukey HSD,FWER=0.05

=============================================================================================
=
            group1                          group2              meandiff  lower  upper  rejec
t
---------------------------------------------------------------------------------------------
-
MCAR_da_median_absolute_error   MCAR_rf_median_absolute_error   -1.9269  -8.0576 4.2039 False

MCAR_da_median_absolute_error MCAR_simple_median_absolute_error -2.7676  -8.8984 3.3631 False

MCAR_rf_median_absolute_error MCAR_simple_median_absolute_error -0.8408  -6.9715  5.29  False

---------------------------------------------------------------------------------------------
```

6.6.2 MAR: One-Way ANOVA and Tukey's HSD Test Results

Figure 6.4 contains the results of Levenes test, ANOVA test(if null hypothesis of Levenes test accepted), Kruskal-Wallis H-test(if null hypothesis of Levenes test rejected), and Tukey

HSD test(if ANOVA test or Kruskal-Wallis H-test found significant differences) of MCAR. It is supplementary to the section 4.2.2.

Figure 6.4: MAR: One-Way ANOVA and Tukey's HSD Test Results

```
accuracy
Levene test
LeveneResult(statistic=29.31970310384083, pvalue=3.87030181671191e-11)
F_onewayResult
F_onewayResult(statistic=44.14059884172583, pvalue=3.549948671982988e-15)
KruskalResult
KruskalResult(statistic=37.207501247441435, pvalue=8.327099641245096e-09)
         Multiple Comparison of Means - Tukey HSD,FWER=0.05
=============================================================
     group1           group2       meandiff  lower   upper  reject
-------------------------------------------------------------
MAR_da_accuracy   MAR_rf_accuracy   -0.0066  -0.0826  0.0693 False
MAR_da_accuracy MAR_simple_accuracy -0.2637  -0.3396 -0.1878  True
MAR_rf_accuracy MAR_simple_accuracy -0.2571   -0.333 -0.1811  True
-------------------------------------------------------------


f1_score
Levene test
LeveneResult(statistic=29.31970310384083, pvalue=3.87030181671191e-11)
F_onewayResult
F_onewayResult(statistic=44.14059884172583, pvalue=3.549948671982988e-15)
KruskalResult
         Multiple Comparison of Means - Tukey HSD,FWER=0.05
=============================================================
     group1           group2       meandiff  lower   upper  reject
-------------------------------------------------------------
MAR_da_f1_score   MAR_rf_f1_score   -0.0066  -0.0826  0.0693 False
MAR_da_f1_score MAR_simple_f1_score -0.2637  -0.3396 -0.1878  True
MAR_rf_f1_score MAR_simple_f1_score -0.2571   -0.333 -0.1811  True
-------------------------------------------------------------


rmse
Levene test
LeveneResult(statistic=0.42243679303074355, pvalue=0.6564433642782659)
F_onewayResult
F_onewayResult(statistic=0.4276951737287202, pvalue=0.6530253650233337)
KruskalResult
KruskalResult(statistic=10.145082644628076, pvalue=0.006266474756073845)
     Multiple Comparison of Means - Tukey HSD,FWER=0.05
============================================================
   group1        group2     meandiff  lower    upper  reject
------------------------------------------------------------
MAR_da_rmse   MAR_rf_rmse   -8.9483  -48.4194 30.5227 False
MAR_da_rmse MAR_simple_rmse  6.3553  -33.1157 45.8264 False
MAR_rf_rmse MAR_simple_rmse 15.3037  -24.1674 54.7747 False
------------------------------------------------------------


median_absolute_error
Levene test
LeveneResult(statistic=1.325845251045372, pvalue=0.26953851697726416)
F_onewayResult
F_onewayResult(statistic=1.3254992810194726, pvalue=0.26962971828189725)
KruskalResult
KruskalResult(statistic=31.18987803698831, pvalue=1.687345581746218e-07)
                     Multiple Comparison of Means - Tukey HSD,FWER=0.05

=============================================================================================
=
           group1                         group2              meandiff  lower    upper  rejec
t
---------------------------------------------------------------------------------------------
-
MAR_da_median_absolute_error   MAR_rf_median_absolute_error   -2.3156  -19.7449 15.1137 False

MAR_da_median_absolute_error MAR_simple_median_absolute_error  8.998   -8.4313  26.4273 False

MAR_rf_median_absolute_error MAR_simple_median_absolute_error 11.3136  -6.1157  28.7429 False

---------------------------------------------------------------------------------------------
```

6.6.3 MNAR: One-Way ANOVA and Tukey's HSD Test Results

Figure 6.5 contains the results of Levenes test, ANOVA test(if null hypothesis of Levenes test accepted), Kruskal-Wallis H-test(if null hypothesis of Levenes test rejected), and Tukey

HSD test(if ANOVA test or Kruskal-Wallis H-test found significant differences) of MNAR. It is supplementary to the section 4.2.2.

Figure 6.5: MNAR: One-Way ANOVA and Tukey's HSD Test Results

```
accuracy
Levene test
LeveneResult(statistic=1.0802219731177067, pvalue=0.34279827116305156)
F_onewayResult
F_onewayResult(statistic=0.8362037804761656, pvalue=0.43586165671445054)
KruskalResult
KruskalResult(statistic=1.9446148943420385, pvalue=0.37820933242205407)
         Multiple Comparison of Means - Tukey HSD,FWER=0.05
====================================================================
     group1             group2        meandiff  lower  upper  reject
--------------------------------------------------------------------
MNAR_da_accuracy   MNAR_rf_accuracy   -0.0212  -0.0606 0.0181 False
MNAR_da_accuracy MNAR_simple_accuracy -0.0079  -0.0473 0.0314 False
MNAR_rf_accuracy MNAR_simple_accuracy  0.0133  -0.0261 0.0526 False
--------------------------------------------------------------------

f1_score
Levene test
LeveneResult(statistic=1.0802219731177065, pvalue=0.34279827116305156)
F_onewayResult
F_onewayResult(statistic=0.8362037804761651, pvalue=0.43586165671445054)
KruskalResult
         Multiple Comparison of Means - Tukey HSD,FWER=0.05
====================================================================
     group1             group2        meandiff  lower  upper  reject
--------------------------------------------------------------------
MNAR_da_f1_score   MNAR_rf_f1_score   -0.0212  -0.0606 0.0181 False
MNAR_da_f1_score MNAR_simple_f1_score -0.0079  -0.0473 0.0314 False
MNAR_rf_f1_score MNAR_simple_f1_score  0.0133  -0.0261 0.0526 False
--------------------------------------------------------------------

rmse
Levene test
LeveneResult(statistic=0.7405686158665844, pvalue=0.47906438879327073)
F_onewayResult
F_onewayResult(statistic=0.7416094842483066, pvalue=0.47857223667087223)
KruskalResult
KruskalResult(statistic=1.557190082644638, pvalue=0.4590505054465117)
      Multiple Comparison of Means - Tukey HSD,FWER=0.05
============================================================
   group1         group2      meandiff  lower    upper  reject
------------------------------------------------------------
MNAR_da_rmse   MNAR_rf_rmse   -8.7988  -38.6692 21.0716 False
MNAR_da_rmse MNAR_simple_rmse -15.2638 -45.1342 14.6067 False
MNAR_rf_rmse MNAR_simple_rmse -6.4649  -36.3354 23.4055 False
------------------------------------------------------------

median_absolute_error
Levene test
LeveneResult(statistic=0.3462325717025468, pvalue=0.7080702218198442)
F_onewayResult
F_onewayResult(statistic=0.34623257170254695, pvalue=0.7080702218198442)
KruskalResult
KruskalResult(statistic=10.853586156111795, pvalue=0.004397174611536061)
                     Multiple Comparison of Means - Tukey HSD,FWER=0.05

=============================================================================================
==
            group1                         group2              meandiff  lower   upper  reje
ct
---------------------------------------------------------------------------------------------
--
MNAR_da_median_absolute_error   MNAR_rf_median_absolute_error    -1.001  -9.0464  7.0445 Fals
e
MNAR_da_median_absolute_error MNAR_simple_median_absolute_error -2.7837  -10.8292 5.2618 Fals
e
MNAR_rf_median_absolute_error MNAR_simple_median_absolute_error -1.7827  -9.8282  6.2628 Fals
e
---------------------------------------------------------------------------------------------
```

6.6.4 Real: One-Way ANOVA and Tukey's HSD Test Results

Figure 6.6 contains the results of Levenes test, ANOVA test(if null hypothesis of Levenes test accepted), Kruskal-Wallis H-test(if null hypothesis of Levenes test rejected), and Tukey

HSD test(if ANOVA test or Kruskal-Wallis H-test found significant differences) of MCAR. It is supplementary to the section 4.2.2.

Figure 6.6: Real: One-Way ANOVA and Tukey's HSD Test Results

```
accuracy
Levene test
LeveneResult(statistic=1.9410343386526725, pvalue=0.14803616550884563)
F_onewayResult
F_onewayResult(statistic=1.4483775770481424, pvalue=0.23902855268021675)
KruskalResult
KruskalResult(statistic=2.911395694543899, pvalue=0.2332375429126299)
         Multiple Comparison of Means - Tukey HSD,FWER=0.05
=====================================================================
     group1             group2        meandiff  lower  upper  reject
---------------------------------------------------------------------
Real_da_accuracy   Real_rf_accuracy    0.0002  -0.0332 0.0337 False
Real_da_accuracy Real_simple_accuracy  0.0209  -0.0126 0.0544 False
Real_rf_accuracy Real_simple_accuracy  0.0207  -0.0128 0.0541 False
---------------------------------------------------------------------

f1_score
Levene test
LeveneResult(statistic=1.9410343386526725, pvalue=0.14803616550884563)
F_onewayResult
F_onewayResult(statistic=1.448377577048142, pvalue=0.23902855268021675)
KruskalResult
         Multiple Comparison of Means - Tukey HSD,FWER=0.05
=====================================================================
     group1             group2        meandiff  lower  upper  reject
---------------------------------------------------------------------
Real_da_f1_score   Real_rf_f1_score    0.0002  -0.0332 0.0337 False
Real_da_f1_score Real_simple_f1_score  0.0209  -0.0126 0.0544 False
Real_rf_f1_score Real_simple_f1_score  0.0207  -0.0128 0.0541 False
---------------------------------------------------------------------

rmse
Levene test
LeveneResult(statistic=0.5038137681266337, pvalue=0.6055266695744359)
F_onewayResult
F_onewayResult(statistic=0.5057889040192076, pvalue=0.6043420644309336)
KruskalResult
KruskalResult(statistic=3.2827685950413183, pvalue=0.19371170197356058)
      Multiple Comparison of Means - Tukey HSD,FWER=0.05
=====================================================================
   group1         group2      meandiff  lower    upper  reject
---------------------------------------------------------------------
Real_da_rmse   Real_rf_rmse   -2.6127  -22.5851 17.3596 False
Real_da_rmse Real_simple_rmse -8.2759  -28.2482 11.6965 False
Real_rf_rmse Real_simple_rmse -5.6631  -25.6355 14.3092 False
---------------------------------------------------------------------

median_absolute_error
Levene test
LeveneResult(statistic=35.32466119425702, pvalue=9.955736266704955e-13)
F_onewayResult
F_onewayResult(statistic=25.180581810014967, pvalue=8.036625628510465e-10)
KruskalResult
KruskalResult(statistic=54.282834483674, pvalue=1.6316697452610085e-12)
                     Multiple Comparison of Means - Tukey HSD,FWER=0.05

=============================================================================================
==
            group1                          group2             meandiff  lower   upper  reje
ct
---------------------------------------------------------------------------------------------
--
Real_da_median_absolute_error   Real_rf_median_absolute_error   -0.0062  -0.0086 -0.0038  Tru
e
Real_da_median_absolute_error Real_simple_median_absolute_error -0.0062  -0.0085 -0.0038  Tru
e
Real_rf_median_absolute_error Real_simple_median_absolute_error  0.0001  -0.0023  0.0024 Fals
e
---------------------------------------------------------------------------------------------
```

References

[All] Paul D Allison. "Multiple Imputation for Missing Data: A Cautionary Tale." p. 14.

[ALR17] Olanrewaju Akande, Fan Li, and Jerome Reiter. "An Empirical Comparison of Multiple Imputation Methods for Categorical Data." *The American Statistician*, **71**(2):162–170, April 2017.

[BLS17] Brett K. Beaulieu-Jones, Daniel R. Lavage, John W. Snyder, Jason H. Moore, Sarah A. Pendergrass, and Christopher R. Bauer. "Characterizing and Managing Missing Structured Data in Electronic Health Records." July 2017.

[Bre] Leo Breiman. "Statistical Modeling: The Two Cultures." *THE TWO CULTURES*, p. 33.

[EB01] Craig Enders and Deborah Bandalos. "The Relative Performance of Full Information Maximum Likelihood Estimation for Missing Data in Structural Equation Models." *Structural Equation Modeling: A Multidisciplinary Journal*, **8**(3):430–457, July 2001.

[ESG] Shahram Ebadollahi, Jimeng Sun, David Gotz, Jianying Hu, Daby Sow, and Chalapathy Neti. "Predicting Patients Trajectory of Physiological Data using Temporal Trends in Similar Patients: A System for Near-Term Prognostics." p. 5.

[GR15] Chandan Gautam and Vadlamani Ravi. "Data imputation via evolutionary computation, clustering and a neural network." *Neurocomputing*, **156**:134–142, May 2015.

[GW17] Lovedeep Gondara and Ke Wang. "MIDA: Multiple Imputation using Denoising Autoencoders." *arXiv:1705.02737 [cs, stat]*, May 2017. arXiv: 1705.02737.

[Comment: To appear in the proceedings of the 22nd Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD 2018).]

[Hip12] Paul T. von Hippel. "Maximum likelihood multiple imputation: Faster, more efficient imputation without posterior draws." *arXiv:1210.0870 [stat]*, October

2012. arXiv: 1210.0870.

[Comment: 36 pages, 6 tables, 1 figure.]

[HKS17] Jianglin Huang, Jacky Wai Keung, Federica Sarro, Yan-Fu Li, Y.T. Yu, W.K. Chan, and Hongyi Sun. "Cross-validation based K nearest neighbor imputation for software quality datasets: An empirical study." *Journal of Systems and Software*, **132**:226–252, October 2017.

[JTS17] Natasha Jaques, Sara Taylor, Akane Sano, and Rosalind Picard. "Multimodal autoencoder: A deep learning approach to filling in missing sensor data and enabling better mood prediction." pp. 202–208. IEEE, October 2017.

[LLK14] Serena G Liao, Yan Lin, Dongwan D Kang, Divay Chandra, Jessica Bon, Naftali Kaminski, Frank C Sciurba, and George C Tseng. "Missing value imputation in high-dimensional phenomic data: imputable or not, and how?" *BMC Bioinformatics*, **15**(1), December 2014.

[SB12] D. J. Stekhoven and P. Buhlmann. "MissForest–non-parametric missing value imputation for mixed-type data." *Bioinformatics*, **28**(1):112–118, January 2012.

[SE17] A.S. Salama and O.G. El-Barbary. "Topological approach to retrieve missing values in incomplete information systems." *Journal of the Egyptian Mathematical Society*, **25**(4):419–423, October 2017.

[STB18] Benjamin Shickel, Patrick Tighe, Azra Bihorac, and Parisa Rashidi. "Deep EHR: A Survey of Recent Advances in Deep Learning Techniques for Electronic Health Record (EHR) Analysis." *IEEE Journal of Biomedical and Health Informatics*, **22**(5):1589–1604, September 2018. arXiv: 1706.03446.

[Comment: Accepted for publication with Journal of Biomedical and Health Informatics: http://ieeexplore.ieee.org/abstract/document/8086133/.]

[Sto] Christine P Stone. "A Glimpse at EHR Implementation Around the World: The Lessons the US Can Learn." p. 12.

[WSH18] Wei Wang, Jack Smith, Hussein Hejase, and Kevin J Liu. "Non-parametric and semi-parametric support estimation using SEquential RESampling random walks on biomolecular sequences." June 2018.

[WZ17] Marvin N. Wright and Andreas Ziegler. "ranger: A Fast Implementation of Random Forests for High Dimensional Data in C++ and R." *Journal of Statistical Software*, **77**(1), 2017. arXiv: 1508.04409.

[YSK18] Pranjul Yadav, Michael Steinbach, Vipin Kumar, and Gyorgy Simon. "Mining Electronic Health Records (EHRs): A Survey." *ACM Computing Surveys*, **50**(6):1–40, January 2018.

[ZW11] Di Zhao and Chunhua Weng. "Combining PubMed knowledge and EHR data to develop a weighted bayesian network for pancreatic cancer prediction." *Journal of Biomedical Informatics*, **44**(5):859–868, October 2011.