Imputation Of Missing Values In Clinical Data

Bachelor’s Thesis in Bioinformatics for Faculty of Engineering and Natural Sciences,
Johannes Kepler University

Author:
Micha Johannes Birklbauer

Supervisors:
Assoz.Univ.-Prof. Dipl.-Ing. Dr. Ulrich Bodenhofer
Hubert Ramsauer, MSc

Linz June 2019

Bibliographical Data

Birklbauer, M. J., 2019: Imputation Of Missing Values In Clinical Data. Bc. Thesis, in English. - 40 p., Faculty of Engineering and Natural Sciences, Johannes Kepler University, Linz, Austria.

Imputation of missing data is a crucial step in data analysis since many statistical methods require complete datasets. In that regard MissForest imputation is a powerful tool that seems to outperform most other imputation approaches. This analysis evaluates how good imputation using MissForest is compared to other methods like imputation by Multivariate Imputation by Chained Equations (MICE), Restricted Boltzmann Machines (RBM) or the standard strawman (mean) imputation in a clinical dataset that is used to predict the mortality of patients after heart valve surgery.

23rd August 2019 [illegible]
Date Signature

Table of Contents

Abstract

Motivation: Imputation of missing data is a crucial step in data analysis since many statistical methods require complete datasets. In that regard MissForest imputation is a powerful tool that seems to outperform most other imputation approaches. This analysis evaluates how good imputation using MissForest is compared to other methods like imputation by Multivariate Imputation by Chained Equations (MICE), Restricted Boltzmann Machines (RBM) or the standard strawman (mean) imputation in a clinical dataset that is used to predict the mortality of patients after heart valve surgery.

Result: Models which were trained on datasets imputed by MissForest, MICE or RBM managed to outperform those which used standard strawman imputation in a most of the cases. The best values for area under the (ROC-)curve and balanced accuracy were achieved using MICE and RBM respectively while strawman imputation still yielded the best result in terms of accuracy. However, the differences were rather small and significant in only 5/66 of the tested cases. The expectation that MissForest, MICE and RBM would overall significantly outperform strawman imputation could therefore not be met. Without a complete reference dataset the true imputation error cannot be calculated and the reason for this outcome can only be speculated upon. Two of the probable causes would be that the missing values are not significant for prediction and that data is not missing at random which negatively affects imputation by MissForest, MICE and RBM.

1. Introduction

Missing data presents a problem in nearly all of today's bigger research projects that depend on big datasets. Missingness may occur in different forms: Not missing at random data may introduce systematic error and make the study population less representative of the general population, while on the other hand missing at random data leads to a significant loss in statistical power and predictive ability. (Waljee AK, Mukherjee A, Singal AG, et al., 2013, p. 1) Many established methods of analysis require fully observed datasets without any missing values, therefore imputation of missing values is a crucial step in data analysis. Many popular imputation methods depend on tuning parameters or specification of a parametric model. The choice of these tuning parameters or models without prior knowledge is a difficult task and might have dramatic effects on a method’s performance. Furthermore a majority of imputation methods are restricted to one type of variable and make assumptions about the distribution of the data or subsets of the variables, which lead to questionable situations e. g. assuming normal distributions. MissForest is an imputation approach based on random forests and is able to handle continuous as well as categorical data while making as few assumptions about structural aspects of the data as possible. (Stekhoven DJ, Bühlmann P, 2011, p. 1f)

Waljee et al., Tang and Ishwaran, Stekhoven and Bühlmann all concluded that MissForest was able to compete with or outperform other imputation methods on their used datasets. (Waljee AK, Mukherjee A, Singal AG, et al., 2013, p. 1) (Tang F, Ishwaran H, 2017, p. 1) (Stekhoven DJ, Bühlmann P, 2011, p. 1) In respect to that, this paper tries to assess if their success is also reproducible in a clinical dataset composing information about heart valve surgery patients. It also compares MissForest imputation to other popular approaches like Multivariate Imputation by Chained Equations, often abbreviated as MICE, and imputation by Restricted Boltzmann Machines (RBM), which have successfully been used as generative models for different types of data in the past. (Hinton G, 2010, p. 3)

The goal was to show if there is a significant difference in prediction accuracy of mortality of patients that underwent heart valve surgery when using MissForest, MICE, RBM or strawman imputation.

2. Methods

The following data and methods were used as part of my analysis.

2.1. Predictive Modeling Of Mortality After Heart Valve Surgeries

"Predictive Modeling Of Mortality After Heart Valve Surgeries" (Bodenhofer U, Haslinger-Eisterer B, Minichmayer A, et al., 2016) is a workflow written in the programming language "R" (R Core Team, 2019) by my supervisor Ulrich Bodenhofer to model post-operative mortality of patients that undergo heart valve surgery.

The workflow consists of:

- The "Heart Valve Surgery Dataset".
- The preprocess-function that does strawman (mean) imputation (STRAW).
- A function that builds a random forest model that is optimized for accuracy (ACC).
- A function that builds a random forest model that is optimized for the area under the ROC-curve (AUC).
- A function that builds a random forest model that is optimized for balanced accuracy (BACC).
- A function that performs double 5-fold cross-validation and evaluates the models.

(Bodenhofer U, Haslinger-Eisterer B, Minichmayer A, et al., 2016, p. 2-6)

2.2. Setup

The idea was to only alter the imputation method implemented in the preprocessing-function and not touch any other part of the "Predictive Modeling Of Mortality After Heart Valve Surgeries"-workflow and compare the results.

Therefore the workflow for this analysis could be described as the following:

- Implement a "missForest" version of the preprocessing-function using MissForest imputation (MF). It should be mentioned here that the notation missForest is used when the

"R"-package is addressed while on the other hand MissForest refers to the general imputation approach. Although these are practically interchangeable.

- Implement a "MICE" version of the preprocessing-function using Multivariate Imputation by Chained Equations (MICE).
- Implement an "RBM" version of the preprocessing-function using Restricted Boltzmann Machines for imputation (RBM).
- Run the "Predictive Modeling Of Mortality After Heart Valve Surgeries"-workflow for the four different imputation methods and different thresholds of missing data. To be more specific the thresholds were chosen in similar fashion to the approaches used by Tang and Ishwaran (Tang F, Ishwaran H, 2017, p. 9) and Waljee, et al. (Waljee AK, Mukherjee A, Singal AG, et al., 2013, p. 4) and increased stepwise by 10% starting from 15% to 35%. One run with high amount of missing data was also performed for each imputation method and in this case a maximum of 60% missingness was chosen. This resulted in runs with maximum 15%, 25%, 35% and 60% missingness respectively. The exact values are chosen arbitrary however and one could also do 10%, 20%, 30% and 70% for example and still should come to the same conclusion!
- Compare the results of models that used MissForest/MICE/RBM imputation to the output of models that used strawman imputation.

2.2.1 Example Workflow of one Experiment

For better understanding let experiments be denoted as **EXP(missingness, imputation method, optimization)** with the following values:

- **missingness**:
  - - : variables that contain missing values are excluded
  - 0.15 : maximum 15% missingness per variable
  - 0.25 : maximum 25% missingness per variable
  - 0.35 : maximum 35% missingness per variable
  - 0.60 : maximum 60% missingness per variable
  - * : all except "-"

- **imputation method**:
    - - : no imputation
    - STRAW : strawman imputation
    - MF : MissForest imputation
    - MICE: MICE imputation
    - RBM: RBM imputation
    - * : all except “-“
- **optimization**:
    - ACC : accuracy optimization
    - AUC : area under the ROC-curve optimization
    - BACC : balanced accuracy optimization
    - * : all

An example workflow then would look like this:

- Choose a maximum amount of missingness (0.15, 0.25, 0.35 or 0.60).
- Choose an imputation method (STRAW, MF, MICE or RBM).
- Choose which value you want to optimize with the random forest (ACC, AUC, BACC).

Choosing 15% maximum missingness, MF imputation and ACC optimization – so for **EXP(0.15, MF, ACC)** - the workflow would continue as the following:

- Exclude all variables from the dataset that contain more that 15% missing values.
- Impute the dataset with MissForest imputation.
- Train the random forests to maximize ACC.
- Perform cross-validation.

Each experiment results in an **AROC.Mean**, an **ACC.Mean** and a **BACC.Mean** from the cross-validation which can be seen in Fig. 3-14.

More examples using the EXP-notation:

- **EXP(*, MF, ACC)** would denote all experiments that used MissForest imputation and accuracy optimization. Therefore four experiments could be denoted by this expression, namely **EXP(0.15, MF, ACC)**, **EXP(0.25, MF, ACC)**, **EXP(0.35, MF, ACC)** and **EXP(0.60, MF, ACC)**.

- **EXP(*, MF, *)** would denote all experiments that used MissForest imputation which would be 12 in total.
- **EXP(*, *, *)** would denote all experiments without **EXP(-, -, *)** which would be 48 in total (all experiments that used imputation).

2.3. Heart Valve Surgery Dataset

The used dataset was composed of data about patients undergoing heart valve surgery. Each patient was represented in a separate row in the dataset and the columns linked to information like laboratory data, test results and clinical annotations.

A snapshot of the demographic and preoperative data of the patients included in the dataset:

Age in years:	mean: 68.0	standard deviation: ±12.6
Height in cm:	mean: 169	standard deviation: ±11
Weight in kg:	mean: 78.0	standard deviation: ±16.1
Sex:	male: 1356 (60.8%)	female: 873 (39.2%)
Patients on dialysis: 1.3%		
Patients with endocarditis: 5.1%		
Patients with cerebrovascular disease: 11.6%		
Patients with peripheral vascular disease: 8%		
Patients with angina pectoris: 26.2%		
Patients with arrhythmia: 22.7%		
Patients with cardiogenic shock: 0.44%		
NYHA-Classification:		
	I / II / III / IV	
(in %)	9.6 / 59.9 / 28.2 / 2.3	
ASA-Classification:		
	I / II / III / III / IV / V	
(in %)	0.1 / 4.9 / 80.9 / 13.9 / 0.2	
Hemoglobin in g/dl:	mean: 13.1	standard deviation: ±1.8
Potassium in mmol/l:	mean: 4.4	standard deviation: ±0.5
Creatinin in mg/dl:	mean: 1.1	standard deviation: ±0.9
BUN in mg/dl:	mean: 21.7	standard deviation: ±11.2
Glomerular filtration rate in ml/min/1.73m^2:		
	mean: 74	standard deviation: ±26
Leukocytes in G/l:	mean: 8.0	standard deviation: ±3.3
Bilirubin in ml/dl:	mean: 0.9	standard deviation: ±0.6
apTT in sec:	mean: 30.9	standard deviation: ±4.5

(Bodenhofer U, Haslinger-Eisterer B, Minichmayer A, et al., 2016, p. 3)

The dataset contains 2229 observations (patients/rows) with 147 variables (columns), only 13 of them are complete and 134 of these variables have missing values. There are zero complete observations, so there was no patient that had a complete record. A visual representation of the missingness in the dataset is shown in the following graph.

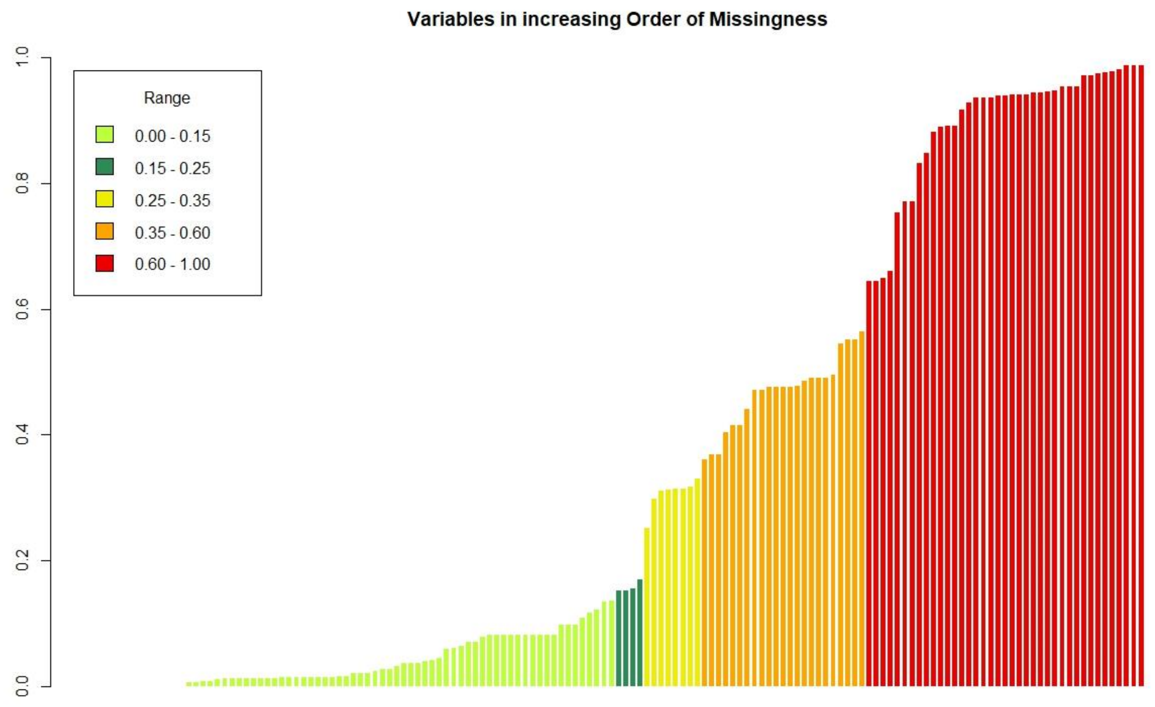

Fig. 1: All variables in the dataset sorted according to their amount of missingness. Each bar represents one variable (13 of them are not shown as bars as they are complete and therefore have 0% missingness). The variables are coloured corresponding to which group they belong to, green means not more than 15% missingness and therefore these variables (73 in total) will be used in every run. Dark green is every variable between 15% and 25% missingness (4 in total). Yellow is every variable in the range from 25% to 35% which amounts to 8 variables. Variables that had more than 35% but less than 60% missingness (23) were coloured orange. The part in red was discarded as those had more than 60% missingness (39 variables) and as described in **2.2 SETUP** there was no run that used variables with more than 60% missingness.

2.4. Data Preprocessing using Strawman (Mean) Imputation

In the first approach the dataset was preprocessed using strawman imputation (or more commonly used is the term “mean imputation”) (Tang F, Ishwaran H, 2017, p. 3f). That meant missing variable values were imputed in the following manner.

Considering an *n x p*-dimensional data matrix $X = (X_1, X_2,\dots, X_p)$ with variables containing missing values and choosing an arbitrary variable X_S including missing values, then the observed values (values that weren’t missing) of variable X_S would be denoted by $y^{(S)}_{obs}$ and the missing values of variable X_S would be denoted by $y^{(S)}_{mis}$. The imputation could then be described as:

- Numerical variables: The missing values $y^{(S)}_{mis}$ were imputed by taking the mean of the observable values $y^{(S)}_{obs}$.
- Ordinal variables: The missing values $y^{(S)}_{mis}$ were imputed by taking the value that had the highest frequency among observable values $y^{(S)}_{obs}$.
- Categorical variables: These were first one-hot-encoded and then missing values $y^{(S)}_{mis}$ were imputed using the most frequent value among observable values $y^{(S)}_{obs}$.
- Binary variables: The missing values $y^{(S)}_{mis}$ were imputed by taking the value that had highest frequency among observable values $y^{(S)}_{obs}$.

(Bodenhofer U, Haslinger-Eisterer B, Minichmayer A, et al., 2016, p. 4)

Furthermore variables that exceeded the threshold of 15/25/35/60 percent missing values were discarded as mentioned in **2.2 SETUP**.

2.5. Data Preprocessing using MissForest Imputation

In the second approach the first step was to one-hot-encode categorical variables but not impute anything yet. The whole but still incomplete dataset was then fed to the missForest-function from the “R”-package “missForest” which took care of imputing missing values. (Stekhoven DJ, 2013)

Similar to the strawman approach, variables that exceeded the threshold of 15/25/35/60 percent missing values were discarded.

2.5.1. MissForest

MissForest is an imputation approach that utilizes the flexibility and versatility of random forest models to predict missing values. It thereby creates a random forest model for each variable in the dataset and uses the rest of the variables to predict the missing values for that variable. This is done in a cyclic fashion for all variables and the entire process is iteratively repeated until a stopping criterion is met. Random forest models are able to handle continuous as well as categorical responses, require very little tuning and provide an internally cross-validated error estimate which are only a few of their advantages. (Waljee AK, Mukherjee A, Singal AG, et al., 2013, p. 3)

To go into more detail, here is how the creators of the “missForest” “R”-package, Stekhoven and Bühlmann, describe their approach:

> Assuming that X = (X_1, X_2,…, X_p) is an *n x p*-dimensional data matrix containing missing values we are able to use a random forest to impute the missing data. The random forest algorithm proposed by Breiman would be able to handle missing values by itself if there is a complete response variable. This is done by weighting the frequencies of the observed values in a variable with the random forest proximities after being trained on the initially mean imputed dataset. (Stekhoven DJ, Bühlmann P, 2011, p. 3)
>
> In the missForest approach however, the missing values are predicted by using a random forest trained on the observed parts of the data set. For an arbitrary variable X_S including missing values at entries $i^{(S)}_{mis} \subseteq \{1,\ldots, n\}$ the dataset can be separated into four parts:
>
> 1. The observed values of variable X_S, denoted by $y^{(S)}_{obs}$;
> 2. The missing values of variable X_S, denoted by $y^{(S)}_{mis}$;

3. The variables other than X_S, with observations $i^{(S)}_{obs} = \{1,\dots, n\} \setminus i^{(S)}_{mis}$ denoted by $x^{(S)}_{obs}$;
4. The variabes other than X_S with observations $i^{(S)}_{mis}$ denoted by $x^{(S)}_{mis}$.

(Stekhoven DJ, Bühlmann P, 2011, p. 3)

At the start we make an initial guess for the missing values in X using mean imputation or any other suitable imputation method. Then the variables X_S, $s = 1,\dots, p$ are sorted according to the amount of missing values in increasing order. For each variable X_S in the dataset the missing values are imputed by first fitting a random forest with response $y^{(S)}_{obs}$ (the observed values of X_S) and predictors $x^{(S)}_{obs}$ (all observations (rows) of variables that are not X_S and do NOT have a missing value in X_S). After that the missing values $y^{(S)}_{mis}$ (the missing values in variable X_S) are predicted by applying the trained random forest to $x^{(S)}_{mis}$ (all observations (rows) of variables that are not X_S and DO have a missing value in X_S). This imputation procedure is done in an iterative fashion until a stopping criterion is met. (Stekhoven DJ, Bühlmann P, 2011, p. 3)

The following pseudo algorithm 1 gives a representation of the missForest method.

Algorithm 1 Impute missing values with random forest.

Require: X an *n x p* matrix, stopping criterion γ

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

(Stekhoven DJ, Bühlmann P, 2011, p. 4)

As shown the algorithm halts as soon as the stopping criterion γ is met. In Stekhoven and Bühlmann's implementation this is the case when the difference between the newly imputed data matrix compared to the previous one increases for the first time with respect to both variable types, if present. The difference for the set of continuous variables N is defined as

$$\Delta N = \frac{\sum_{j \in N}(X_{new}^{imp} - X_{old}^{imp})^2}{\sum_{j \in N}(X_{new}^{imp})^2}$$

and for the set of categorical variables F as

$$\Delta F = \frac{\sum_{j \in F} \sum_{i=1}^{n} I_{X_{new}^{imp} \neq X_{old}^{imp}}}{\#NA}$$

where #NA is the number of missing values in the categorical variables.

After imputation of the missing values the performance can be assessed by using the normalised root mean squared error (NRMSE) for continuous variables and by using the proportion of falsely classified entries over categorical missing values for categorical variables. This results in values close to 0 if the performance is good and values close to 1 if the performance is bad. (Stekhoven DJ, Bühlmann P, 2011, p. 4)

2.6. Data Preprocessing using MICE

The third approach was carried out the same way as the imputation with MissForest but instead of "missForest" the "MICE" package was used to impute missing values. (Van Buuren S, Groothuis-Oudshoorn K, 2018) MICE is short for Multivariate Imputation by Chained Equations and the package uses Fully Conditional Specification (FCS) (Van Buuren S, 2012, p. 108) implemented by the MICE algorithm as described by Van Buuren and Groothuis-Oudshoorn (a more detailed explanation can be found below). Again variables that exceeded a certain threshold of missing values were discarded.

2.6.1. Multiple Imputation

Before diving into MICE we should clarify what we mean by "multiple imputation". MICE is a multiple imputation technique which means that after imputing a dataset with missing values we not only get one but multiple "complete" datasets. In comparison to single imputation procedures like the previously used strawman imputation or also MissForest imputation, multiple imputation has a few advantages. For example the analysis of the multiple imputed datasets takes into account the uncertainty in the imputations and yields accurate standard errors due to the multiple predictions for each missing value. To be more specific, this means that if there is not much information in the observed part of the data the imputations will be very variable which leads to a high standard error. On the other hand if the observed part of the data is very descriptive and highly predictive of the missing values the imputed values will be very similar across all imputations which results in a smaller standard error. (Azur MJ, Stuart EA, Frangakis C, et al., 2011, p. 2)

2.6.2. MICE - Multivariate Imputation by Chained Equations

Multivariate Imputation by Chained Equation is an alternative approach to multiple imputation procedures that rely on a large joint model – such as a joint normal distribution – for all of the variables. Especially in large datasets with hundreds of variables of different types such an assumption is rarely appropriate. Because of this, there is a need for other methods like MICE which has proven to produce accurate and useful results in cases that exceed thousands of observations and hundreds of variables. The MICE algorithm runs a series of regression models where each variable with missing data is modelled conditional upon the other variables in the dataset. This means that each variable can be modelled according to its own distribution. (Azur MJ, Stuart EA, Frangakis C, et al., 2011, p. 2)

The chained equation process can be broken down into several general steps:

1. Every missing value in the dataset is imputed by a simple imputation method like strawman imputation. These values however are only place holders.
2. Place holder values are set back to missing for one variable X_S.
3. Observed values $y^{(S)}_{obs}$ from variable X_S are regressed on the on the other variables in the dataset. In other words, X_S is the dependent variable in a regression model and all other remaining variables are independent variables in the regression model. These regression models work the same as when performing linear, logistic or Poison regression outside of the context of missing data.
4. The missing values $y^{(S)}_{mis}$ from the variable X_S are replaced by the predictions from the regression model in Step 3. When the next variable X_T (and all other remaining variables) is imputed, X_S and all its values, hence $y^{(S)}_{obs}$ and the imputed $y^{(S)}_{mis}$, are used as an independent variable in the regression model for X_T.
5. The procedure described in steps 2-4 is then repeated for every variable containing missing values. After iterating over all variables one “cycle” is done. At the end of this “cycle” all of the missing values – that initially were filled by strawman imputation – have been replaced by regressions that reflect the relationships observed in the dataset.
6. Usually not only one but several “cycles” are performed and the missing values are updated after each “cycle”. At the end of the last “cycle” the imputed values are saved and a completed dataset is returned. The number of “cycles” is flexible and can be specified by the researcher. In general 10 “cycles” are suggested but to identify the optimal number of “cycles” e.g. in identifying when the imputations converge, more research is needed. (Azur MJ, Stuart EA, Frangakis C, et al., 2011, p. 3)

This is the general approach that Multivariate Imputation by Chained Equations follows. The software implementations of MICE differ in some points however. The specific implementation from the “R”-package by Stef van Buuren and Karin Groothuis-Oudshoorn (that was used in this research) as well as a more detailed and mathematical explanation will be shown in the next two subchapters.

2.6.3. Fully Conditional Specification

Fully Conditional Specification (Van Buuren S, 2012, p. 108) or often just abbreviated as FCS is an imputation technique that fills in multivariate missing data on a variable-by-variable basis. It requires the specification of an imputation model for each variable containing missing values and creates imputations per variable in an iterative fashion.

Contrary to traditional joint modelling approaches, FCS specifies the multivariate distribution $P(Y, X, R \mid \theta)$ through a set of conditional densities $P(Y_j \mid X, Y_{-j}, R, \phi_j)$ which is used to impute Y_j given X, Y_{-j} and R. The missing values are first imputed by simple random draws from the distribution and then the imputation by FCS is done by iterating over all variables and their conditionally specified imputation models. Imputation with FCS is a natural generalization of univariate imputation. (Van Buuren S, 2012, p. 108)

In the context of missing data imputation similar terms have been coined describing the same or a similar idea e.g. “chained equations” by Van Buuren and Groothuis-Oudshoorn. (Van Buuren S, 2012, p. 109)

2.6.4. The MICE algorithm and “R”-package by Van Buuren and Groothuis-Oudshoorn

Again considering an $n \times p$-dimensional data matrix $X = (X_1, X_2, \ldots, X_p)$ with variables containing missing values, then:

- X_j would be the j^{th} column of that data matrix,
- X_j^{mis} would be the missing values of variable X_j,
- X_j^{obs} would be the observed values of variable X_j,
- and X_{-j} would be all other columns of data matrix X except j.

Applying that, Van Buuren describes the implemented MICE algorithm as the following:

Algorithm 2 MICE algorithm for imputation of multivariate missing data

```
1. Specify an imputation model $P(X_j^{mis} \mid X_j^{obs}, X_{-j}, R)$ for variable $X_j$ with $j = 1,\dots,p$.
2. For each $j$, fill in starting imputations $X_j^0$ by random draws from $X_j^{obs}$.
3. Repeat for $t = 1,\dots,T$:
4. Repeat for $j = 1,\dots,p$:
5. Define $X_{-j}^t = (X_1^t,\dots, X_{j-1}^t, X_{j+1}^{t-1},\dots, X_p^{t-1})$ as the currently complete data except $X_j$.
6. Draw $\phi_j^t \sim P(\phi_j^t \mid X_j^{obs}, X_{-j}^t, R)$.
7. Draw imputations $X_j^t \sim P(X_j^{mis} \mid X_j^{obs}, X_{-j}^t, R, \phi_j^t)$.
8. End repeat $j$.
9. End repeat $t$.
```

(Van Buuren S, 2012, p. 110; notation adjusted)

Parameter T refers to the number of iterations and can be specified by the user. Usually a number of 5-10 iterations is sufficient. (Van Buuren S, 2012, p. 109)

The name “chained equations” is referring to the fact that the MICE algorithm can easily be implemented as a concatenation of univariate procedures to impute missing values. The *mice()* function in the “R”-package runs the algorithm *m* times in parallel which results in *m* differently imputed datasets. (Van Buuren S, Groothuis-Oudshoorn K, 2011, p. 7)

The MICE package offers the user complete control over the imputation procedure and it is possible to specify a separate univariate imputation model for each column. The default imputation methods are predictive mean matching for numeric data, logistic regression for binary data, polytomous regression for unordered categorical data and a proportional odds model for ordered data with more than two levels. MICE provides many other methods and it is also possible to implement your own imputation functions and use them in the algorithm. (Van Buuren S, Groothuis-Oudshoorn K, 2018, p. 67f)

2.7. Data Preprocessing using Restricted Boltzmann Machines

In the fourth and final approach Restricted Boltzmann Machines (RBM) (Hinton G, 2007, p. 1) were trained to impute the missing values in the dataset. Differently to MICE and similar to MissForest this procedure produces only a single complete dataset (no multiple imputations). Contrary to all other imputation methods, imputation with RBM was carried out in "python" (Python Software Foundation, 2019) instead of "R" since "python" offered a more convenient implementation. I used the package "SherlockML-BoltzmannClean" by ASI Data Science which implements a scikit-learn transformer interface for creating and training a Restricted Boltzmann Machine which can be used to impute a pandas-dataframe. The package follows the guidelines established by Geoffrey Hinton. (ASI Data Science, 2018) Similar to all other imputation approaches variables that exceeded the threshold of 15/25/35/60 percent missing values were discarded.

2.7.1. Restricted Boltzmann Machines

When neuron-like units that make stochastic decisions about whether to be on or off are symmetrically connected to form a network, this network is called a Boltzmann Machine. Boltzmann Machines can be used to solve two different computational problems:

- Firstly, search problems where the weights on the connections are fixed and used to represent a cost function to generate data vectors with low cost values.
- Secondly, learning problems where the data vectors are fixed and used to learn the weights of the connections to generate the data vectors with high probability.

(Hinton G, 2007, p. 1)

Since missing value imputation can be viewed as a learning problem the usage of Boltzmann Machines seems to be a promising approach. However since learning is typically very slow in Boltzmann Machines the deployment of the more efficient Restricted Boltzmann Machines is obvious. (Hinton G, 2007, p. 4) The major differences are that in a Restricted Boltzmann Machine there only are two layers of units, one layer consisting of visible units and one layer of hidden units. Furthermore, unlike in the Boltzmann Machine, not all units are connected with each other, the connections are only between visible-hidden/hidden-visible units which means there are no connections within the layers itself, hence the name “restricted”. This way the hidden units are conditionally independent and learning can be done more efficiently. (Hinton G, 2007, p. 4)

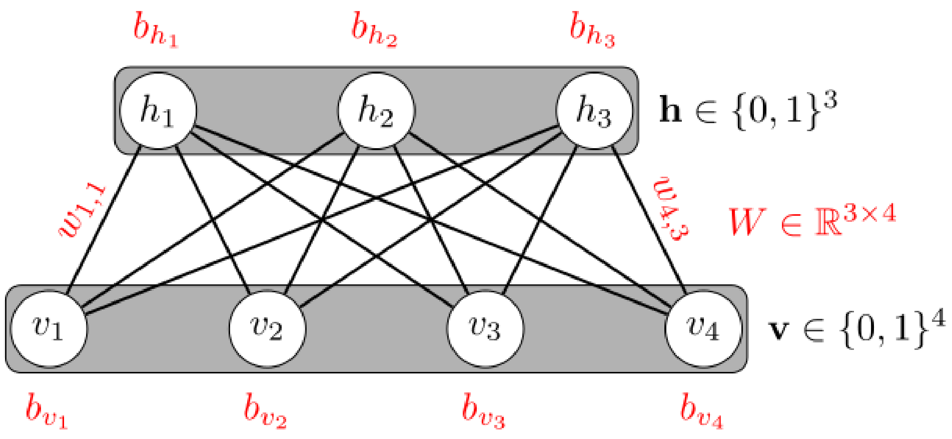

Fig. 2: Example of a Restricted Boltzmann Machine with 4 visible units in the visible layer (v) and 3 hidden units in the hidden layer (h). The learned parameters are coloured in red: Bias vectors for visible and hidden layer (bv, bh) and the connections/weight-matrix (W).

2.7.2. Learning in a Restricted Boltzmann Machine using Contrastive Divergence

To elaborate further on how and what is learned in a Restricted Boltzmann Machine we consider a binary feature vector as input, e.g. it could be the pixels of a black and white image.

Geoffrey Hinton describes the learning procedure (Hinton G, 2010, p. 3ff) as the following:

> According to Hopfield a joint configuration, (***v***, ***h***) of the visible and hidden units has an energy given by:
>
> $$(1) \quad E(v,h) = -\sum_{i \in visible} a_i v_i - \sum_{j \in hidden} b_j h_j - \sum_{i,j} v_i h_j w_{ij}$$
>
> where $\boldsymbol{v_i}$, $\boldsymbol{h_j}$ are the binary states of visible unit ***i*** and hidden unit ***j***, $\boldsymbol{a_i}$, $\boldsymbol{b_j}$ are their biases and $\boldsymbol{w_{ij}}$ is the weight between them. The network assigns a probability to every possible pair of a visible and a hidden vector via this energy function:
>
> $$(2) \quad p(v,h) = \frac{1}{Z} e^{-E(v,h)}$$

where the "partition function", $\boldsymbol{Z}$, is given by summing over all possible pairs of visible and hidden vectors:

$$(3) \quad Z = \sum_{v,h} e^{-E(v,h)}$$

The probability that the networks assigns to a visible vector, ***v***, is given by summing over all possible hidden vectors:

$$(4) \quad p(v) = \frac{1}{Z}\sum_{h} e^{-E(v,h)}$$

The probability that the network assigns to a training vector, or in our case a training image, can be raised by adjusting the weights and biases to lower the energy of that vector/image and raise the energy of other vectors/images, especially those that have low energies and therefore make a big contribution to the partition function. The derivative of the log probability of a training vector with respect to a weight is surprisingly simple and given as the following:

$$(5) \quad \frac{\partial \log p(v)}{\partial w_{ij}} = \langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{model}$$

where the angle brackets are used to denote expectations of the in the subscript specified distribution. This results in a very simple learning rule for performing stochastic steepest ascent in the log probability of the training data:

$$(6) \quad \Delta w_{ij} = \in (\langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{model})$$

where $\in$ is the learning rate.

Because there are no direct connections between hidden units in an RBM, it is very easy to get an unbiased sample of $\langle v_i h_j \rangle_{data}$. Given a randomly selected training vector, ***v***, the binary state, $\boldsymbol{h_j}$, of each hidden unit, ***j***, is set to 1 with probability

$$(7) \quad p\left(h_j = 1 \middle| v\right) = \sigma(b_j + \sum_i v_i w_{ij})$$

where $\sigma(x)$ is the logistic sigmoid function $1/(1 + \exp(-x))$. $\boldsymbol{v_i h_j}$ is then an unbiased sample.

Because there also are no direct connections between visible units in an RBM, it is also very easy to get an unbiased sample of the state of a visible unit, *given a hidden vector*

(8) $p(v_i = 1|h) = \sigma(a_i + \sum_j h_j w_{ij})$

However getting an unbiased sample of $\langle v_i h_j \rangle_{model}$, is a lot more difficult.

Therefore Hinton proposed a much faster learning procedure in 2002 that starts by setting the states of the visible units to any training vector. Secondly the binary states of the hidden units are all computed in parallel using the equation in step 7. Once the binary states have been chosen for the hidden units, a "reconstruction" is produced by setting each $\boldsymbol{v_i}$ to 1 with a probability given by the equation used in step 8. The change in a weight is then given by

(9) $\Delta w_{ij} = \in (\langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{recon})$

A simplified version of the same learning rule that uses the states of individual units instead of pairwise products is used for the biases. (Hinton G, 2010, p. 3ff)

The learning procedure works very well in practice even though it is only crudely approximating the gradient of the log probability of the training data. The learning algorithm is actually much more closely approximating the gradient of another objective function called Contrastive Divergence (hence the name). (Hinton G, 2010, p. 5)

Contrastive Divergence generally provides biased estimates, however the bias is usually very small. Another advantage of Contrastive Divergence learning is that the procedure is relatively fast and therefore it can be used as a first step to get close to a maximum-likelihood solution before another slower maximum-likelihood learning rule fine tunes the Contrastive Divergence model. (Carreira-Perpiñán M, Hinton G, 2005, p. 1)

2.8. Result Analysis

To assess the performance of the different models under given imputation methods I used the following tools and statistical tests (used "R" functions denoted as *name {package}*):

- **T-Test for pairwise comparison of means (*t.test {stats}*).**

  To assess if there is an overall tendency over all the experiments that one or more imputation methods generally significantly perform better than their counterparts, t-tests are used. In these t-tests one sample always consists of either all **AROC.Means**, **ACC.Means** or **BACC.Means** of all experiments of one imputation method e.g. all **EXP(*, STRAW, *)**. Similar the other sample then consists of all values from another imputation method e.g. **EXP(*, MF, *)**. Furthermore this means that each sample consist of 12 values in total.

- **McNemar's Chi-squared Test for pairwise comparisons of ACC (*mcnemar.test {stats}*).**

  The McNemar's Chi-squared Test is used to compare different accuracy optimized models. In this case the predictions from one random forest of one experiment e.g. **EXP(0.15, STRAW, ACC)** are compared to the predictions of another random forest from another experiment e.g. **EXP(0.15, MF, ACC)**. Since the samples are the predictions the sample size is 2229 – the number of patients in our dataset.

- **DeLong Test for pairwise comparison of AUC (*roc.test {pROC}*).**

  The DeLong Test in "R" allows for comparison of the AUC from two ROC-curves. The models that were optimized for area under the ROC-curve are compared using this test. One sample therefore consists of one ROC-curve from one random forest of one experiment e.g. **EXP(0.15, STRAW, AUC)** and the other sample of another ROC-curve from another random forest of another experiment e.g. **EXP(0.15, MF, AUC)**.

3. Results

The following results were achieved with the imputation methods and the workflow described in the previous chapter. This chapter is subdivided into sections for each tested amount of missing values, a short performance overview and an in-depth analysis at the end that clarifies if and which of the differences are significant or not.

As a short recap of what is mentioned in **2.1 PREDICTIVE MODELING OF MORTALITY AFTER HEART VALVE SURGERIES** and in **2.2 SETUP** the following plots show the means of the results of 5-fold cross-validation setups where models have been built with either 15, 25, 35 or 60 percent maximum missingness and have been optimized to either maximize AUC, ACC or BACC (see plot description which threshold and optimization was used).

3.1. Maximum 15% Missingness

Results for a maximum of 15% missing values per column in the dataset. To rehearse, that means that only 73 out of all 147 variables were used.

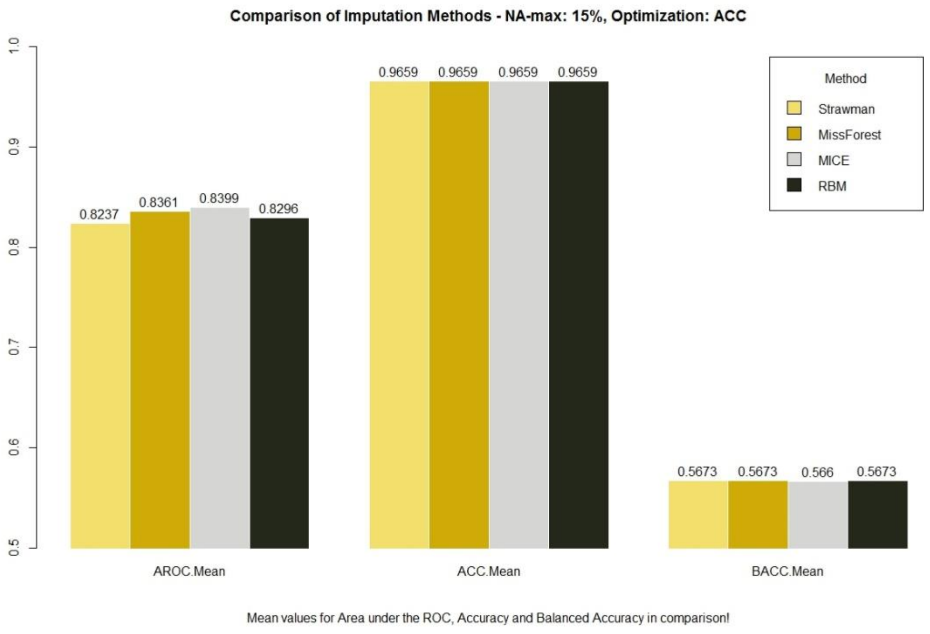

Fig. 3: Comparison of the four different imputation methods when up to 15% missing values are allowed and the model is optimized to maximize accuracy. MICE has the highest AUC with 0.8399, while there is no difference between the different methods in ACC. MICE performs slightly worse than the other methods in terms of BACC. The plot shows results from **EXP(0.15, *, ACC)**.

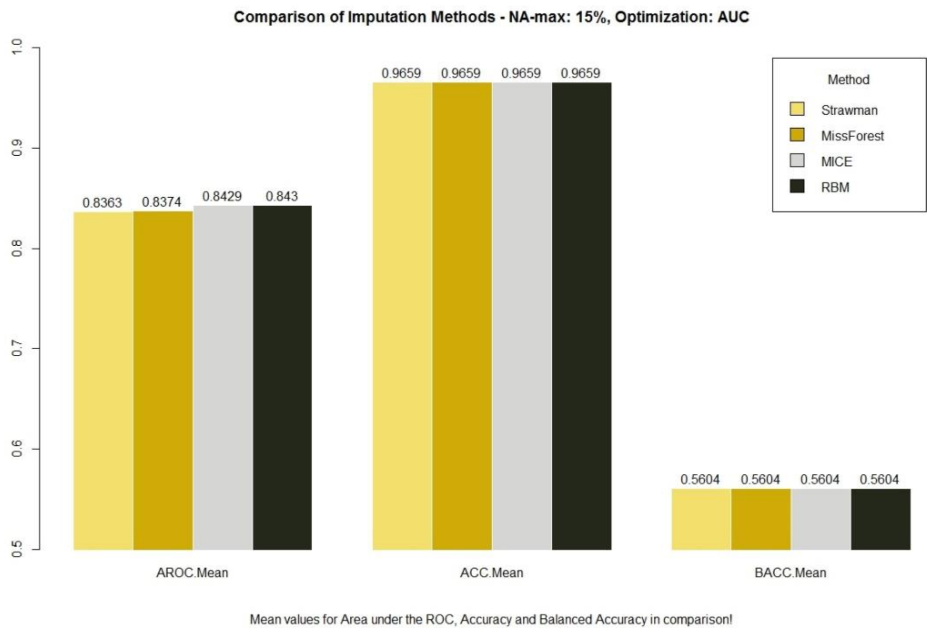

Fig. 4: Comparison of the four different imputation methods when up to 15% missing values are allowed and the model is optimized to maximize the area under the ROC-curve. RBM has the highest AUC with 0.843, while there is no difference between the different methods in ACC and BACC. The plot shows results from **EXP(0.15, *, AUC)**.

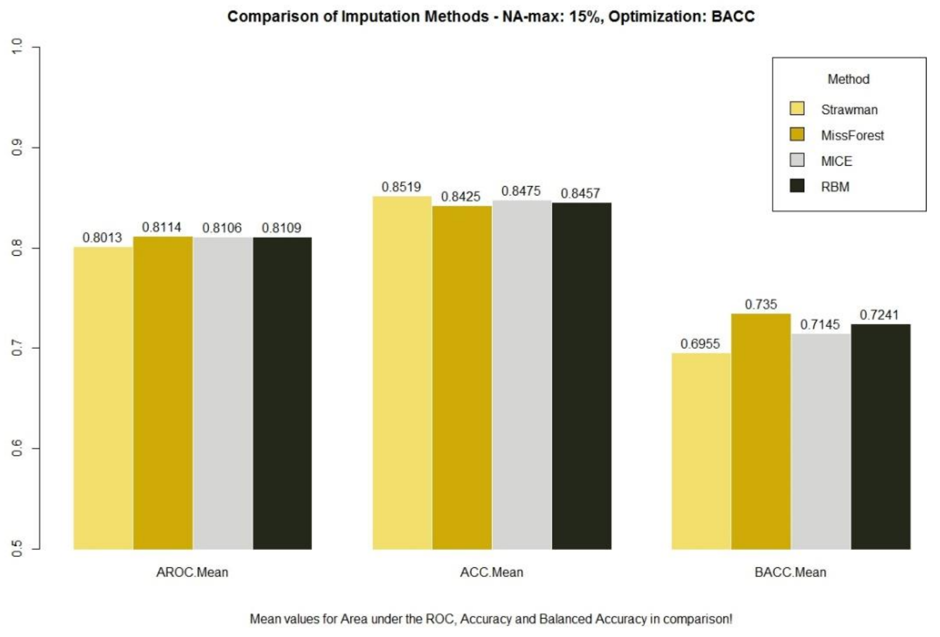

Fig. 5: Comparison of the four different imputation methods when up to 15% missing values are allowed and the model is optimized to maximize balanced accuracy. MF has the highest AUC with 0.8114, while STRAW leads in ACC with 0.8519. MF also outperformed all other methods in terms of BACC with 0.735 balanced accuracy. The plot shows results from **EXP(0.15, *, BACC)**.

3.2. Maximum 25% Missingness

Results for a maximum of 25% missing values per column in the dataset. Rising the threshold to 25% meant an increase of 4 variables and therefore a total of 77 (out of 147) variables that were used.

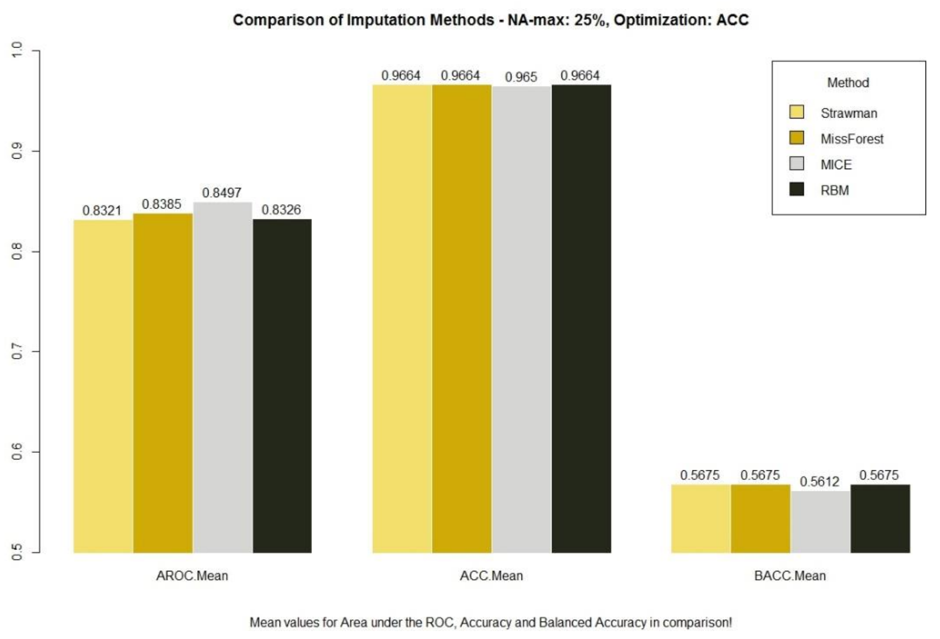

Fig. 6: Comparison of the four different imputation methods when up to 25% missing values are allowed and the model is optimized to maximize accuracy. MICE has the highest AUC with 0.8497, while it performs slightly worse than the other methods when it comes to ACC and BACC. The plot shows results from **EXP(0.25, *, ACC)**.

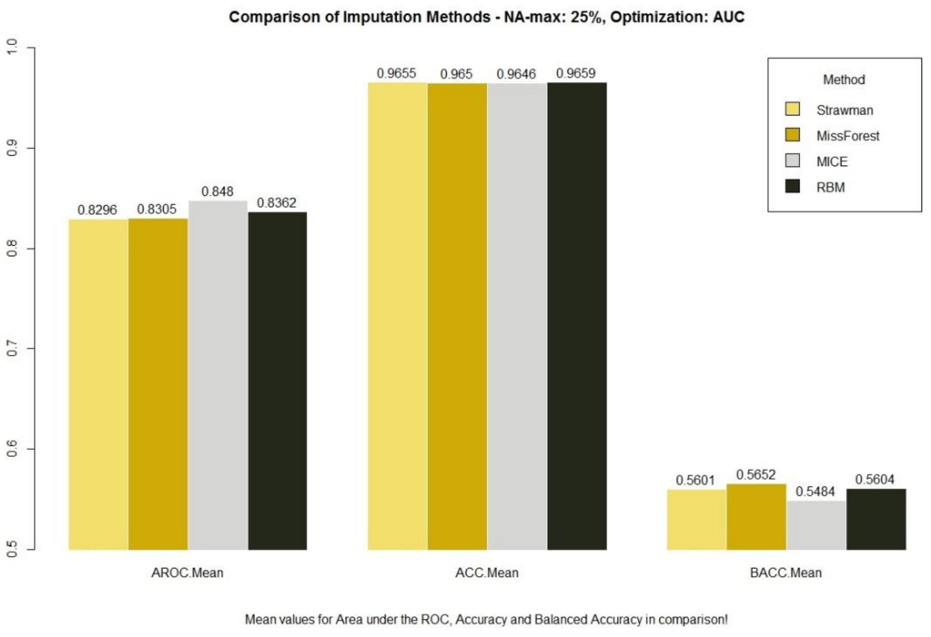

Fig. 7: Comparison of the four different imputation methods when up to 25% missing values are allowed and the model is optimized to maximize the area under the ROC-curve. MICE has the highest AUC with 0.848, while there is almost no difference between the methods in ACC (RBM is slightly ahead). MF performs the best in terms of BACC with 0.5652 balanced accuracy. The plot shows results from **EXP(0.25, *, AUC)**.

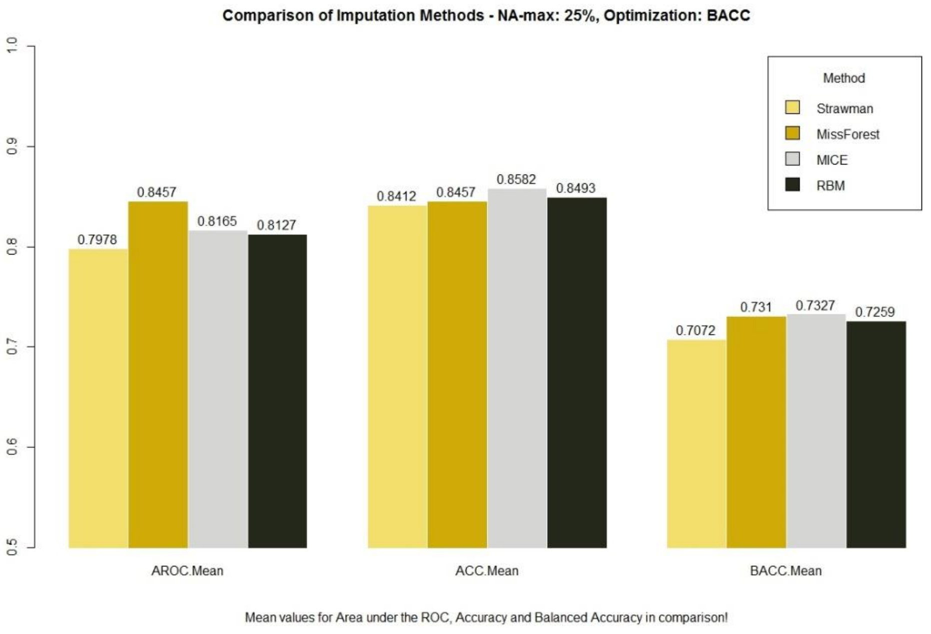

Fig. 8: Comparison of the four different imputation methods when up to 25% missing values are allowed and the model is optimized to maximize balanced accuracy. MF has the highest AUC with 0.8457, while MICE leads in ACC and BACC with 0.8582 and 0.7327 respectively. The plot shows results from **EXP(0.25, *, BACC)**.

3.3. Maximum 35% Missingness

Results for a maximum of 35% missing values per column in the dataset. In total 85 out of the 147 variables were used which meant a net gain of 8 variables compared to 25% maximum missingness.

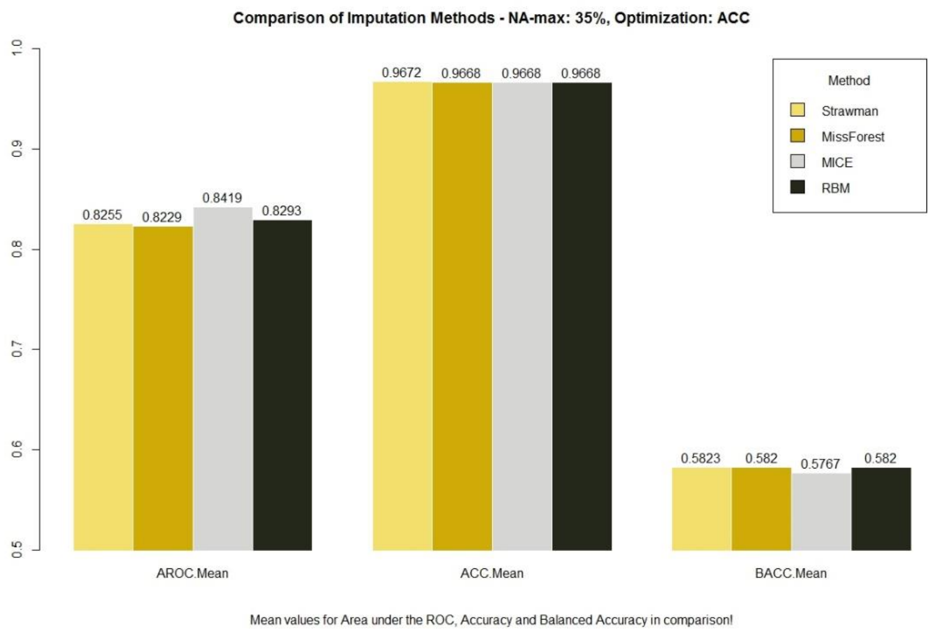

Fig. 9: Comparison of the four different imputation methods when up to 35% missing values are allowed and the model is optimized to maximize accuracy. MICE has the highest AUC with 0.8419, while strawman imputation leads in both ACC and BACC with 0.9672 and 0.5823 respectively. The plot shows results from **EXP(0.35, *, ACC)**.

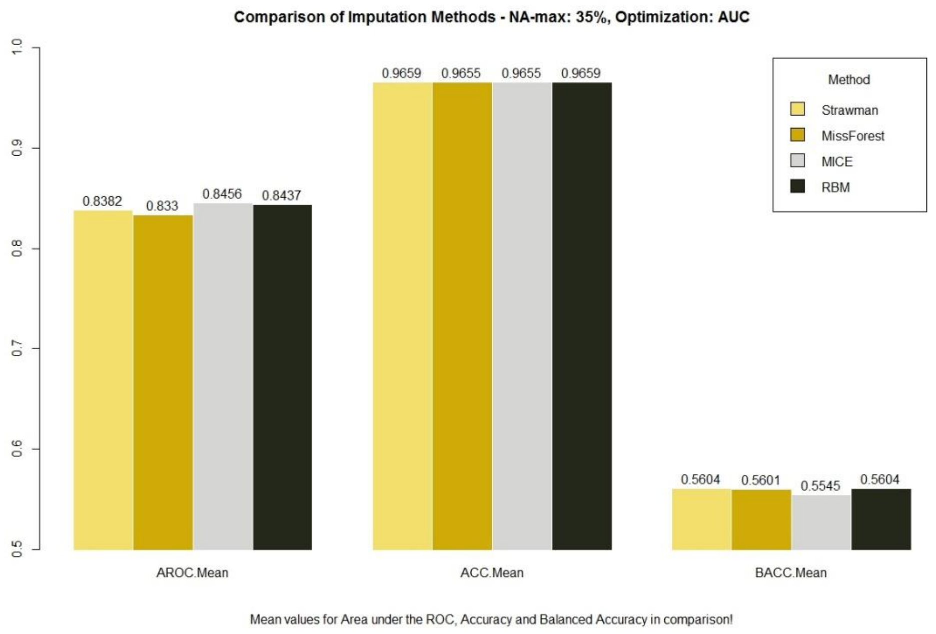

Fig. 10: Comparison of the four different imputation methods when up to 35% missing values are allowed and the model is optimized to maximize the area under the ROC-curve. MICE has the highest AUC with 0.8456, while both strawman imputation and RBM achieve the same, best results in ACC and BACC with 0.9659 and 0.5604. The plot shows results from **EXP(0.35, *, AUC)**.

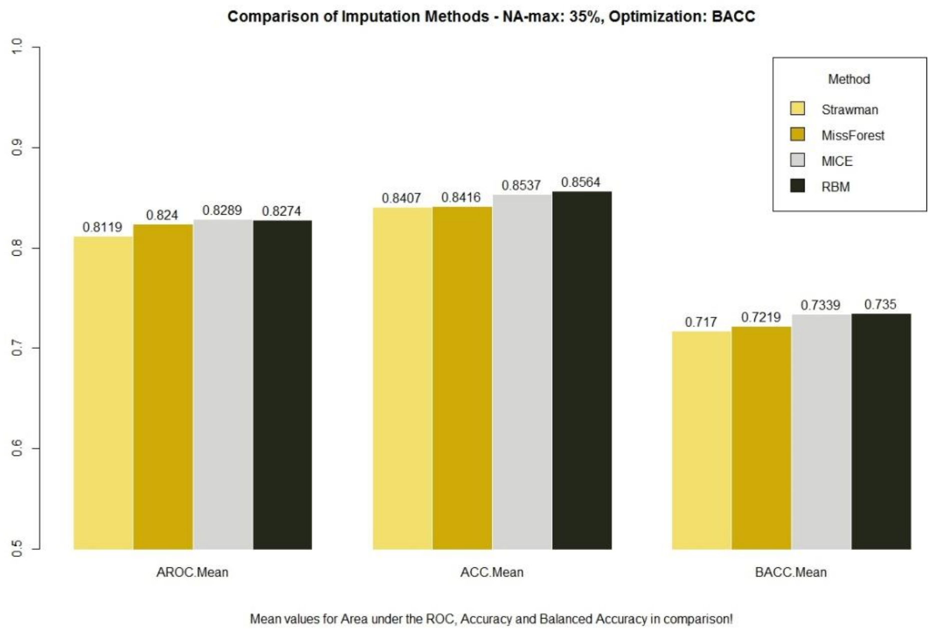

Fig. 11: Comparison of the four different imputation methods when up to 35% missing values are allowed and the model is optimized to maximize balanced accuracy. MICE has the highest AUC with 0.8289, while RBM leads in both ACC and BACC with 0.8564 and 0.735 respectively. The plot shows results from **EXP(0.35, *, BACC)**.

3.4. Maximum 60% Missingness

Results for a maximum of 60% missing values per column in the dataset. Extending to a maximum amount of 60% missingness meant a further addition of 23 variables, resulting in 108 out of the 147 variables to be used.

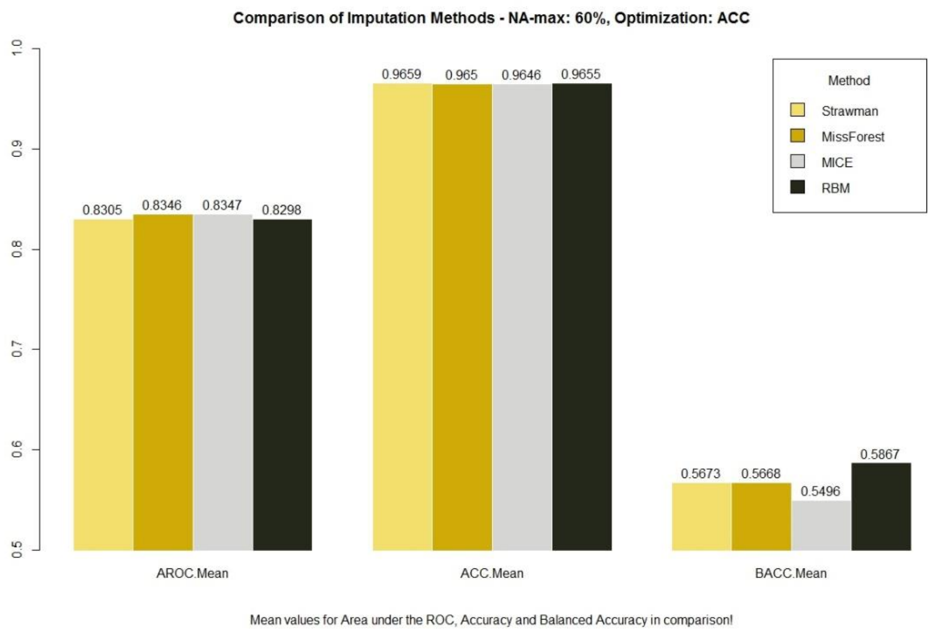

Fig. 12: Comparison of the four different imputation methods when up to 60% missing values are allowed and the model is optimized to maximize accuracy. MICE has the highest AUC with 0.8347, while strawman imputation leads in ACC with 0.9659 and RBM outperforms it's counterparts in BACC with 0.5867 balanced accuracy. The plot shows results from **EXP(0.60, *, ACC)**.

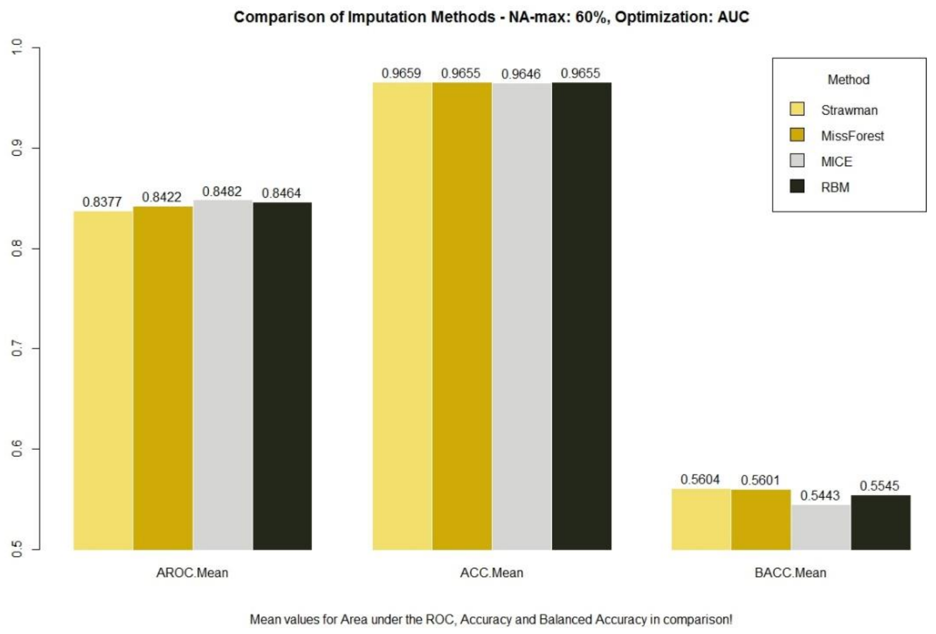

Fig. 13: Comparison of the four different imputation methods when up to 60% missing values are allowed and the model is optimized to maximize the area under the ROC-curve. MICE has the highest AUC with 0.8482, while strawman imputation leads both in terms of ACC and BACC with 0.9659 and 0.5604 respectively. The plot shows results from **EXP(0.60, *, AUC)**.

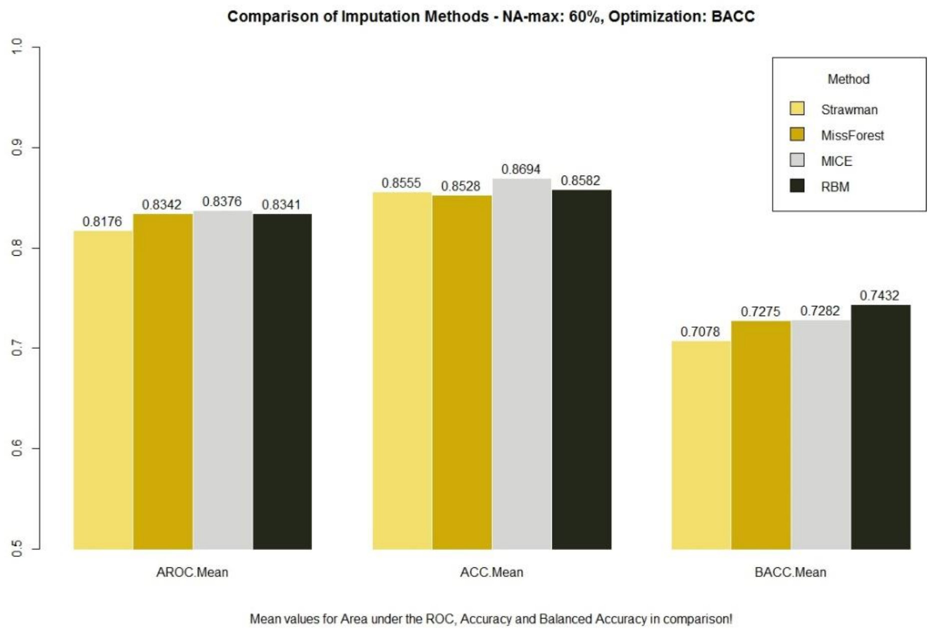

Fig. 14: Comparison of the four different imputation methods when up to 60% missing values are allowed and the model is optimized to maximize balanced accuracy. MICE has the highest AUC with 0.8376 and ACC with 0.8694. RBM leads in terms of BACC with 0.7432 balanced accuracy. The plot shows results from **EXP(0.60, *, BACC)**.

3.5. Performance without Imputation

In order to get an ever wider view of the performance, one run without the imputation of missing values was done. This means that all columns that had missing values somewhere were completely discarded from the dataset and only the complete variables (13 in total) were used for building the model. The result can be seen in the next plot:

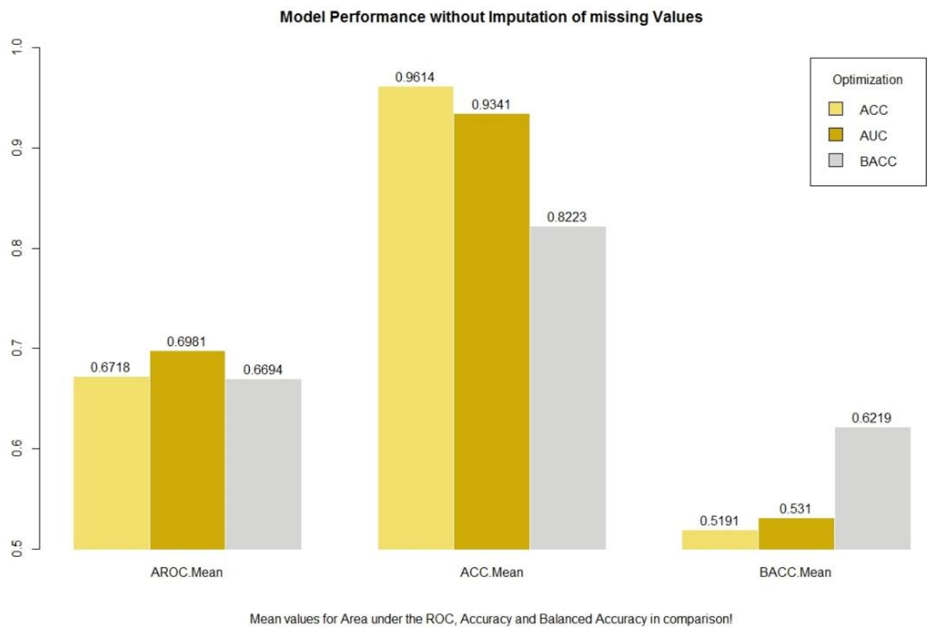

Fig. 15: Model performance in comparison of the three different optimization methods. As one would expect, AUC optimization yields the best AUC, ACC optimization the best ACC and BACC optimization the best BACC. All values are smaller than any of the imputed counterparts however! The plot shows results from **EXP(-, -, *)**.

3.6. Performance Overview

There is no general answer to what method performed best without doing further analysis. However it can be assessed directly from the results in which cases each method produced the best values and how often. The results also immediately show which models produced the overall best values in regards of AUC, ACC and BACC, namely:

- Best AUC: Imputation with MICE, maximum of 25% missing values, ACC optimization → 0.8497 AUC.
- Best ACC: Imputation with strawman method, maximum of 35% missing values, ACC optimization
  → 0.9672 ACC.
- Best BACC: Imputation with RBM, maximum of 60% missing values, BACC optimization → 0.7432 BACC.

3.7. Analysis

Further in-depth analysis is needed to address if MissForest, MICE and RBM significantly outperformed standard strawman imputation. For that matter the tools described in the last chapter in **2.8 RESULT ANALYSIS** were used and produced the following results. For better comparison the fields of significant p-values were coloured in green and the significantly better performing method was denoted in bold green.

The null hypothesis H_0 is that there is no difference between the compared imputation methods.

The alternative hypothesis H_1 is that there is a difference between the compared imputation methods.

Furthermore a significance level of 5% is chosen ($\alpha = 0.05$).

Compared Methods	Compared Values	Test	P-Value
MF, STRAW	ACC	T-Test	0.9746
MF, STRAW	AUC	T-Test	0.07471
MF, STRAW	BACC	T-Test	0.8063
MICE, STRAW	ACC	T-Test	0.9032
MICE, STRAW	AUC	T-Test	0.01961
MICE, STRAW	BACC	T-Test	0.9642
RBM, STRAW	ACC	T-Test	0.9453
RBM, STRAW	AUC	T-Test	0.14
RBM, STRAW	BACC	T-Test	0.7621
MF, MICE	ACC	T-Test	0.8775
MF, MICE	AUC	T-Test	0.3314
MF, MICE	BACC	T-Test	0.8552
MF, RBM	ACC	T-Test	0.9197
MF, RBM	AUC	T-Test	0.7705
MF, RBM	BACC	T-Test	0.9551
MICE, RBM	ACC	T-Test	0.958
MICE, RBM	AUC	T-Test	0.2478
MICE, RBM	BACC	T-Test	0.8135

Fig. 16: Pairwise comparison of means of the different imputation methods to assess if one method generally outperforms the others. Significant p-values are coloured in green. MICE generally performed significantly better than strawman imputation in terms of AUC. As an example, the first line shows the comparison of **ACC.Means** yielded by all models that used MF – so **EXP(*, MF, *)** – **(sample 1)** to all models that used STRAW – so **EXP(*, STRAW, *)** – **(sample 2)**. To be more specific that means that sample 1 consisted of all MissForest imputed ACC.Mean values from Fig. 3-14 which leads to in total 12 values per sample. The same concept holds for sample 2.

Compared Methods	Compared Values	Missingness	Test	P-Value
MF, STRAW	ACC	0.15	McNemar	1
MF, STRAW	ACC	0.25	McNemar	1
MF, STRAW	ACC	0.35	McNemar	1
MF, STRAW	ACC	0.60	McNemar	1
MICE, STRAW	ACC	0.15	McNemar	1
MICE, STRAW	ACC	0.25	McNemar	1
MICE, STRAW	ACC	0.35	McNemar	0.2207
MICE, STRAW	ACC	0.60	McNemar	0.02334
RBM, STRAW	ACC	0.15	McNemar	NA
RBM, STRAW	ACC	0.25	McNemar	1
RBM, STRAW	ACC	0.35	McNemar	1
RBM, STRAW	ACC	0.60	McNemar	1
MF, MICE	ACC	0.15	McNemar	1
MF, MICE	ACC	0.25	McNemar	1
MF, MICE	ACC	0.35	McNemar	0.1306
MF, MICE	ACC	0.60	McNemar	0.04123
MF, RBM	ACC	0.15	McNemar	1
MF, RBM	ACC	0.25	McNemar	NA
MF, RBM	ACC	0.35	McNemar	1
MF, RBM	ACC	0.60	McNemar	0.6831
MICE, RBM	ACC	0.15	McNemar	1
MICE, RBM	ACC	0.25	McNemar	1
MICE, RBM	ACC	0.35	McNemar	0.1306
MICE, RBM	ACC	0.60	McNemar	0.01333

Fig. 17: Pairwise comparison of ACC optimized results using the McNemar-test to assess if there are significant differences in these specific scenarios. Significant p-values are coloured in green. MICE performed significantly better than all other methods in cases where high amounts of missing data was permitted (maximum 60% missing values). As an example, the first line shows the comparison of results from the ACC optimized random forest trained with the MF imputed dataset and a maximum of 15% missing values – so **EXP(0.15, MF, ACC)** – **(sample 1)** to the ACC optimized random forest trained with the STRAW imputed dataset and a maximum of 15% missing values – so **EXP(0.15, STRAW, ACC)** – **(sample 2)**. In more detail that means that sample 1 consisted of all 2229 predictions from the random forests that were trained to maximize accuracy on the dataset that had a maximum of 15% missingness per variable and was imputed with MissForest. Again the same concept for sample 2.

Compared Methods	Compared Values	Missingness	Test	P-Value
MF, STRAW	AUC	0.15	DeLong	0.3375
MF, STRAW	AUC	0.25	DeLong	0.05335
MF, STRAW	AUC	0.35	DeLong	0.1914
MF, STRAW	AUC	0.60	DeLong	0.3173
MICE, STRAW	AUC	0.15	DeLong	1
MICE, STRAW	AUC	0.25	DeLong	0.1469
MICE, STRAW	AUC	0.35	DeLong	0.1549
MICE, STRAW	AUC	0.60	DeLong	0.3173
RBM, STRAW	AUC	0.15	DeLong	1
RBM, STRAW	AUC	0.25	DeLong	1
RBM, STRAW	AUC	0.35	DeLong	1
RBM, STRAW	AUC	0.60	DeLong	0.3173
MF, MICE	AUC	0.15	DeLong	0.3375
MF, MICE	AUC	0.25	DeLong	0.01405
MF, MICE	AUC	0.35	DeLong	0.05083
MF, MICE	AUC	0.60	DeLong	0.3272
MF, RBM	AUC	0.15	DeLong	0.3375
MF, RBM	AUC	0.25	DeLong	0.5335
MF, RBM	AUC	0.35	DeLong	0.1914
MF, RBM	AUC	0.60	DeLong	0.3375
MICE, RBM	AUC	0.15	DeLong	1
MICE, RBM	AUC	0.25	DeLong	0.1469
MICE, RBM	AUC	0.35	DeLong	0.1549
MICE, RBM	AUC	0.60	DeLong	0.5652

Fig. 18: Pairwise comparison of AUC optimized results using DeLong-test to assess if there are significant differences in these specific scenarios. Significant p-values are coloured in green. MissForest performed significantly better than MICE in the case of 25% missing values at maximum. As an example, the first line shows the comparison of results from the AUC optimized random forest trained with the MF imputed dataset and a maximum of 15% missing values – so **EXP(0.15, MF, AUC)** – **(sample 1)** to the AUC optimized random forest trained with the STRAW imputed dataset and a maximum of 15% missing values – so **EXP(0.15, STRAW, AUC)** – **(sample 2)**. For the DeLong-test that means that sample 1 was the ROC-curve produced by the random forests that were trained to maximize AUC on the dataset that had a maximum of 15% missingness per variable and was imputed with MissForest. Again the same concept for sample 2.

3.8. Computational Speed and Implementation Effort

Excluding strawman imputation the fastest method definitely was MICE which imputed the whole dataset in roughly a few minutes. RBM is a little behind MICE and imputation took up to half an hour. The worst of the methods definitely was MissForest which took about 20 minutes per iteration for 100 trees per forest increasing to 7 hours per iteration for 10 000 trees per forest. On the other hand MissForest can easily be run in parallel which can drastically speed up the imputation time if one has the necessary resources. One has to also note here, that this dataset is rather small and therefore the time for imputation is still acceptable. In bigger setups this might actually be an aspect to think of however.

For reference: The used computer was equipped with an intel Core i7-4790K processor (CPU) running at 4GHz and 16 GB DDR3-2133MHz random access memory (RAM).

In terms of implementation effort the most easy to use method was MissForest as the “R”-package is really straight forward to use and well documented. MICE is slightly less convenient since the package is a lot mightier and offers a lot of tools for multiple imputation that not directly are involved with the MICE imputation method itself, which can be slightly confusing. The least convenient method was imputation by RBM as there is not really any material available in “R” and one has to either implement it from scratch or use one of the less known packages where it is unsure how trustworthy they are. For that step “python” was the programming language of choice and there is an imputation package available which does RBM imputation given a pandas-dataframe. This was mostly inconvenient because of the communication of values between “R” and “python” since the preprocessing is first done with “R”, the values have then to be transferred to “python” were the missing ones are imputed by the RBM and afterwards the values have to be sent back to “R” to complete the rest of the workflow. If the workflow was completely written in “python” this might have been a lot more straight forward and easier to implement.

4. Interpretation of Results and Discussion

MissForest and MICE imputation were both praised by Waljee, Mukherjee, Singal, et al. as well as Tang and Ishwaran to be a promising method for imputation of missing values that does a significantly better job than other imputation methods including strawman imputation, therefore the question arises why the results in our dataset and setup are not significantly improving across most cases regardless if imputed by MissForest, MICE or RBM. To rehearse, there were only 5 out of 66 tested cases where imputation significantly improved the results.

Since we don’t have a complete reference dataset we can’t determine the true imputation error and we are left guessing why the results are actually how they are. There are various possible reasons that could cause this behaviour:

- The values that are imputed don’t affect the prediction in a significant way.
- Strawman imputation might already give good predictions for missing values.
- Data might not be missing at random which negatively affects MissForest, MICE and RBM imputation.

Most likely the observed behaviour is caused by a combination of all three points. Notable is that the imputed variables are indeed affecting the prediction which is shown by results of the run without imputation (significantly worse accuracy, balanced accuracy and area under the curve). However the difference in amount of variables between using only complete variables (13) and variables up to 15% missingness (73) is pretty big. Concluding from that we can assume that the variables with missing values are important and significantly improve the result, but on the other hand the 15% that is missing in those variables might not be.

Concerning the last point, it seems also very likely that data is not missing at random since clinical tests are usually taken based on what other tests have yet been performed and not been performed, which leads to missing variables (lab data) that is both dependent on observed and missing values. (Tang F, Ishwaran H, 2017, p. 9)

5. Conclusion

In conclusion MissForest, MICE and RBM manage to outperform standard strawman imputation in a most of the cases. The best values for AUC and BACC were achieved using MICE and RBM respectively while strawman imputation still yielded the best result in terms of ACC. In our setup however the differences were rather small and significant in only 5/66 of the tested cases. The expectation that MissForest, MICE and RBM would significantly outperform strawman imputation across the field could therefore not be met. Since there is no complete reference dataset the true imputation error cannot be calculated and the cause for this behaviour can only be speculated upon. The most probable reasons are that missing values are not significant for the prediction and that the data in the dataset is not missing at random which negatively affects imputation by MissForest, MICE and RBM.

6. Acknowledgements

This work was supported foremost by my supervisors Dipl.-Ing. Dr. Ulrich Bodenhofer and Hubert Ramsauer, MSc; my University Johannes Kepler Universität Linz and the Johannes Kepler Universitätsklinikum who provided the data.

7. List of Figures

8. References

ASI Data Science. Package “sherlockml-boltzmannclean”: Fill missing values in a Pandas DataFrame using a Restricted Boltzmann Machine. PyPI 2018. Available from: https://pypi.org/project/sherlockml-boltzmannclean/;

Azur MJ, Stuart EA, Frangakis C, et al. Multiple Imputation by Chained Equations: What is it and how does it work? International Journal of Methods in Psychiatric Research 2011;

Bodenhofer U, Haslinger-Eisterer B, Minichmayer A, Hermanutz G, and Meier J. Machine Learning-Based Risk Profile Classification of Patients Undergoing Heart Valve Surgery. Johannes Kepler University and Kepler University Clinic 2016;

Breiman L. Random forests. University of California 2001;

Carreira-Perpiñán MA, Hinton G. On Contrastive Divergence Learning. University of Toronto 2005;

Hinton G. Boltzmann Machine. Scholarpedia 2007;

Hinton G. A Practical Guide to Training Restricted Boltzmann Machines. University of Toronto 2010;

Oba S, Sato M, Takemasa I, Monden M, Matsubara K, and Ishii S. A Bayesian missing value estimation method for gene expression profile data. PubMed 2003;

Python Software Foundation. The Python Programming Language. USA: 2019. Available from: https://www.python.org/;

R Core Team. R: The R Project for Statistical Computing. Vienna, Austria: 2019. Available from: https://www.R-project.org/;

Stekhoven DJ. Package “missForest”: Nonparametric Missing Value Imputation using Random Forest. CRAN 2013; / Daniel J. Stekhoven (2013). missForest: Nonparametric Missing Value Imputation using Random Forest. R package version 1.4. Available from: https://cran.r-project.org/package=missForest;

Stekhoven DJ, Buehlmann P. MissForest - nonparametric missing value imputation for mixed-type data. Bioinformatics 2012;

Tang F, Ishwaran H. Random Forest Missing Data Algorithms. University of Miami 2017;

Van Buuren S. Flexible Imputation of Missing Data. CRC Press 2012;

Van Buuren S, Groothuis-Oudshoorn K. Package “mice”: Multivariate Imputation by Chained Equations. CRAN 2018. Available from: https://cran.r-project.org/package=mice;

Van Buuren S, Groothuis-Oudshoorn K. mice: Multivariate Imputation by Chained Equations in R. Journal of Statistical Software 2011;

Waljee AK, Mukherjee A, Singal AG, et al. Comparison of imputation methods for missing laboratory data in medicine. BMJ Open 2013;