BACHELOR

A Research of Model Combinations in Multiply Robust Imputation

van Lieshout, Tjerk J.

Award date:
2023

A Research of Model Combinations in Multiply Robust Imputation

A thesis by Tjerk van Lieshout

A bachelor final project guided by:
An-Chiao Liu and Ton de Waal
Bachelor Data Science Final Project

Technical University of Eindhoven

June 2023

Abstract

Multiply robust imputation is an imputation method that attempts to properly predict values for missing data by employing the use of multiple classification methods at once. This approach is based on the notion that including more methods increases the chances of properly capturing the real context of the data one is working with.

Selecting which imputation methods to combine is an important factor in the performance of multiply robust imputation. This thesis looks at an example where models of different natures are combined to find out more about whether they will work with or against one another when trying to impute missing categorical variables.

An experiment is conducted using three different data sets to compare the performance between a multiply robust imputation method that employs only logistic regression models and a multiply robust imputation method that employs a mix of logistic regression models and random forest models.

In the experiment it was demonstrated that the two models showed cooperative behavior, which increased their performance compared to their performance in isolation.

Contents

1 Introduction

1.1 Missing Data Imputation and the Existing Methods

When working with data from the outside world, it is common to discover missing data in the form of missing entries or the absence of specific variables for certain entries. This can be caused due to human errors or equipment malfunctions. There are three main reasons as to why data sets can end up containing missing data. Data is missing as it was forgotten or lost, the data might not have been applicable to the specific entry, or the data is missing as it is to no interest of the specific entry. Missing data can be dealt with in two ways. Firstly, one can discard the incomplete or missing entries in a data set This will likely reduce the amount of information one is able to retrieve from the data set, as the amount of entries is reduced.The second way to deal with missing data is by replacing any missing values with estimated values, which are obtained using the intact observations from the data set. This method of recovering a data set is known as imputation. This can only be done for entries where only a couple of variables are missing however. Replacing an entire missing entry is of course not possible using estimation methods as there is no available information about this entry to use for comparisons. [SAPV19].
The distinction between whether data is missing due to identifiable or unidentifiable reasons is important to make. If data is missing for identifiable reasons, imputing values may add bias to the data set, making it unrecoverable. Data imputation can be of great value when working with data that is missing for unidentifiable reasons however. This may seem counter intuitive, but if the reason for missing data is unidentifiable, this is associated with data missing at random, meaning that imputation is less likely to introduce a bias. This makes data sets with unidentifiable missing data, so-called recoverable data sets [SAPV19].
There are many different types of imputation methods. In general, most methods follow either machine learning model principles or standard statistic theory. Imputation methods can fall into several categories, some of which are shortly explored here as their properties are important for the main topic of this thesis.

Firstly, an imputation method can be parametric or non-parametric, indicating whether they employ the use of values outside of those in the data set itself. An example of a non-parametric model is Nearest-neighbor imputation, which only uses comparisons between values in the data set to see which completed entry most closely resembles the entry with the missing variable. An example of a parametric model is linear regression imputation, which uses a set of weights on observed independent variables to estimate the unobserved dependent variable. An imputation method can also function as either a donor method or as an estimator method. For a donor method, only values that are actually observed in the data set are imputed. This has the added benefit of making sure that all imputed values are all possible to occur, as they have occurred naturally before. Alternatively, an estimator method employs functions to get as close to the ac-

tual answer as possible. Using the aforementioned methods, Nearest-neighbor imputation functions as as a donor method while linear regression imputation is a clear example of an estimator method. [CHS21]
There is also a distinction to be made between single and multiple imputation methods. As the name implies, single imputation methods only impute a single value. While this is computationally efficient, it may be challenging to measure the proper variance of estimates based on the imputed data. Multiple imputation methods generate numerous estimated values for each missing value in the data set. The estimated values are then combined using statistical or machine learning based methods to form the final imputation. An advantage of this method is that measuring the variance of the estimates is a lot easier. A common example of single imputation would be mean imputation, where all missing values are simply replaced by the mean value of that numerical variable in the data set [Li15]. Examples of multiple imputation include doubly robust or multiply robust imputation, the latter of which is the main topic of this thesis and will be expanded upon hereafter [ZHYH17].

1.2 Multiply Robust Imputation

The multiply robust approach is a specific method that uses multiple prediction/classification techniques together in order to prevent model misspecification. Model misspecification means that the model or formula that a prediction technique is based on, does not represent the reality of the context one is working with. This misspecification can cause distortions in the accuracy of prediction results in the form of unpredictable bias.
Multiply robust imputation aims to fix this issue by using multiple prediction strategies. These prediction techniques can be based on both machine learning model principles or standard statistic theory. When multiply robust imputation is executed on a data set with missing values, each prediction model contained in the algorithm comes with its own set of predictions. Afterwards, using weights for each of the predicted results, the final imputations are decided using either a majority vote in the case of equal weights, or the highest sum of weights in case of unequal weights. Since multiple models are at play at once here, the chances of the model containing a correctly specified model that fits the context is drastically increased, thereby decreasing the chance of model misspecification [CHS21].

1.3 Categorical Variable Imputation

When it comes to imputing missing data, working with numerical variables has a substantial benefit over working with categorical variables. Numerical values always have an intrinsic ordering, making comparisons between them easy to execute and understand. Categorical variables often do not have an explicit ordering, meaning that many imputation algorithms relying on this property will fail to work with this data. As an intuitive example, both humans and machines find the difference between two numbers easier to comprehend than

the difference between two colors. This "ordered" nature of numerical variables makes them easier to work with, which has resulted in a wider variety of options regarding machine learning models and statistical methods when it comes to missing data imputation.
The nature of categorical variables also imposes restrictions on the categories in which their imputation methods belong to. For starters, a categorical variable imputation method can only fall in the donor category. Precise estimation, as is possible with numerical values, is not a feasible option for categorical values as they are discrete. Imputations always belong to a discrete group and nowhere in between.
As an exception to the problem of a lack or order, there exist ordinal variables. These are a form of categorical variables which do feature an intrinsic ordering. An example of this would be clothing sizes (S, M, L, XL etc,). Variables such as these can be mapped to numerical values, making them suitable for the aforementioned imputation algorithms that require the property of ordering.
The less intuitive nature of categorical variables make them harder to work with, warranting different approaches then when it comes to numerical variables. Because of this, categorical variable imputation, which is the topic of this thesis, is a significant part of data imputation that warrants its own research.

1.4 The current study

This thesis is focused on the performance of multiple imputation using multiply robust imputation on categorical variables and how this can be effected by the models or functions contained within it. The nature of multiply robust imputation allows for a lot of freedom regarding both which and how many models and functions can be used to arrive at the final imputations. In theory, the more methods included in the algorithm, the less likely it becomes for the results to be biased due to a misspecified model. Including more models also leads to more required computations and a larger runtime however. This makes the selection of which models to include important when it comes to optimizing this imputation method [CHS21].
Regarding, predictions of categorical variables, logistic regression is among the most popular. Unlike many other models, the dependent variable that is being predicted in logistic regression is binary, meaning that the regression returns either true or false for a certain variable [Gui14]. Specifics on why and how logistic regression works well with categorical variables will be expanded upon further in the methods section.
As discussed before, a big risk of working with classification models is model misspecification, meaning that one's models do not properly capture the real context of the data [CHS21]. For the current research, the goal is to find out whether using only logistic regression models has a good chance of properly approximating the real context of the data, or whether achieving this is more likely by including a secondary type of model as well. Preferably one with a vastly different assumption on the context. An attempt is made to find an approximate answer to this by comparing a multiply robust imputation method using only

logistic regression models, to a multiply robust imputation method using both logistic regression models and another popular machine learning method. Imputation techniques will be tested on imputation accuracy and estimator bias. The research question is as follows:

What are the performance-based differences between a multiply robust imputation algorithm using logistic regression models exclusively and one utilizing a combination of logistic regression models and another model type when imputing categorical variables?

Noteworthy is that this research question can and will not be conclusively answered in this thesis. As an infinite amount of model combinations are possible with different parameters and data sets for them to be implemented on, the answer will always be specific to the exact context. This thesis instead, mostly aims to gain new insights about the cooperation of different types of models in the multiply robust imputation method.

After this introduction, a section on machine learning methods regarding categorical variable classification will follow. In this section, logistic regression and three other popular methods will be expanded upon. One of these three other methods will be chosen to work alongside logistic regression in the multiply robust imputation model based on their properties. This decision will be based on their workings, efficiency, transparency and explainability.
The next section describes the experiment between the two multiply robust imputation methods. There will be a subsection about the used tools and methods, the data collection and lastly the results.
The last section will focus on the results from the previous section to form a conclusion. Finally, there will be a quick discussion about the validity of the results and its implications, along with general thoughts about the research as a whole.

2 Machine Learning Model Selection

2.1 Section Overview

As stated in the introduction, this section will contain information about the workings and properties of machine learning models regarding categorical variable classification. Specifically, logistic regression, k-Nearest-neighbors, random forests and gradient boosting. It is important to note that not all of these models are going to be used in the final experiment, meaning that these explanations are mostly here for the reader to re-familiarize themselves with the methods as well as to justify the final model selection choice. The information relevant to this current study will continue at the 'Model Decision' paragraph.
From the last three mentioned models, one will be chosen to work alongside

logistic regression in the research experiment. This decision will be based on the models workings, efficiency, transparency and explainability. The first two factors are important for model performance, as I'd like the algorithms to have similar efficiency, so that the running time of the two models that will be constructed are at least similar. The last two factors are important as they are relevant in many research fields, and since logistic regression is an explainable and transparent prediction, it would be nice for the alternative model to have this feature as well [TL10].
The chosen model will be selected in the last subsection. Even though logistic regression is certainly going to be included in the research experiment, it is still valuable to understand its working for comparisons, final conclusions and discussions.
As previously discussed, logistic regression is chosen due to its high popularity for categorical variable classification. The last three methods are chosen due to the large differences in their workings. Although this is just a hypothesis, one might argue that observing four drastically different methods could give more insight into the workings of categorical variable classification, opposed to only looking at similar approaches. This might help in making the final model decision more guided and thought out.

2.2 Logistic Regression

2.2.1 Technical explanation

Making use of binary variables, or dummy variables as they are sometimes called, is a good way of dealing with categorical variables for both the independent and dependent variables. Logistic regression is an algorithm that aims to determine the value of a dependent binary value. It does this by giving weights to the values of the independent variables, which can be both numerical and categorical in nature. For this model, it is assumed that the dependent variable y follows a Bernoulli distribution. This means that there is a probability $P(y = 1)$ lying between 0 and 1 so that the binary value of y is 1. When the logistic regression function returns a higher or equal value to this probability, the binary value of y is 1. When the function returns a lower value, the binary value of y is 0. This probability can be manually decided [Gui14].
The sigmoid function which decides the probability for a certain entry is composed of all relevant independent variables and their respective weights. Below, the basic structure of the logistic regression function is given.

$$\hat{Y} = \frac{e^{\beta_0 + \sum \beta_j X_{ij}}}{1 - e^{\beta_0 + \sum \beta_j X_{ij}}}$$

The weights for each particular variable are decided by using methods that use statistical principles to estimate the strength of the connection between independent and dependent variables. The exact working of this is not relevant for this study and is therefore not explored further [TL10]. The regression weights

of binary variables, function a little bit differently than the weights of numerical variables.
For numerical values in regression, each variable is given a weight, which is then multiplied with the numerical value to compute the actual value that is entered into the regression function. Below is a quick example, with β_1 as the weight for variable 1 and x_i as the value of variable 1 for entry i.

$$Weight_i = \beta_1 x_i$$

Dummy variables are different in the sense that their weights are either turned on or off since the only possible values of the dummy variables are 0 and 1. Looking back at the weight function above, it is clear that if x_i can only be 0 or 1, the resulting weight is either 0 or β_1 [Alk12].
Since categorical variables only have a discrete amount of values, it is possible to split up a categorical variable into multiple dummy variables. One for each value the original categorical variable could have. This principle is what allows logistic regression to be used for multivariate data imputation. Independent categorical variables can be converted into multiple dummy variables with their own weights. As for the dependent variable, probabilities are computed for each possible outcome and the most likely result is selected to be the final output.

2.2.2 Properties

As previously mentioned, logistic regression assumes that the dependent variable follows a Bernoulli distribution. Logistic regression is a relatively efficient algorithm. Once the regression function is made, all one needs to do for a prediction is fill in the numbers. This, along with logistic regression's transparency and explainability are the main reasons as to why it is such a popular choice in multiple research fields [BSGB22]. In the end, logistic regression is just a function with weights, very transparent. It is also easy to explain to others what factors are being taken into account and to what degree in order to make predictions.

2.3 k-Nearest Neighbors

2.3.1 Technical explanation

As previously mentioned, the k-Nearest Neighbors (KNN) imputation method is a donor method, which means that instead of estimating a value for the dependent variable, an actually observed entry for this variable is chosen as the result instead. This makes KNN a potent choice for predicting categorical variables, as it does not have to give much thought into what sort of variable is being donated. KNN functions in the exact same way regardless if numerical or categorical variables are being predicted [FT22].
KNN selects a donor entry by computing 'distances' between the entry that is being predicted and the other complete entries. To compute these distances, a

vector is created from all relevant independent variables featuring their values. Then, using functions such as Manhattan distance, entries can be compared. It is sometimes also important to give each independent variable a weight, as value gaps may have huge differences. The difference between categorical variables can be computed using several methods. Examples of this would be Simple Matching of Coefficients, which distance is simply 1 when the values differ and 0 if they are the same. There also exists Minkowski difference which splits categorical variables up in multiple dummy variables, each representing a different possible value for the categorical variable. By comparing the dummy variable values of two entries, a matrix can be made which is then used for calculations [FT22].
K-Nearest-Neighbors finds the variable value for an entry by comparing its independent variable values to the entries' k nearest neighbors, meaning the k closest entries distance-wise. When predicting a categorical variable, the final result is selected using a vote, with each neighbor voting for their own value of the independent variable. While it is possible to give each vote from the neighbors the same weight, it is also common practice to give the votes weights based on how close the respective neighbor is [FT22].

2.3.2 Properties

Context-wise, KNN makes the assumption that to find the correct output to an entry, the output must be the same as the complete entry that most closely resembles it.
KNN works well for low-dimensional data, but quickly becomes computationally heavy for high-dimensional data. This is because it is required to compute every distance between one entry and the other complete entries to find the nearest neighbors. When the amount of variables that need to be checked doubles, so does the running time, and it is not uncommon to see data with over a hundred variables. There is research about getting around this issue with solutions like parallel computing however [FT22].
KNN is a very clear and explainable model. Not only can its workings be quickly summarized in a single sentence, understanding the final decision is as simple as looking at the functions and potential weights that are used for computing distances along with what variables have been used.

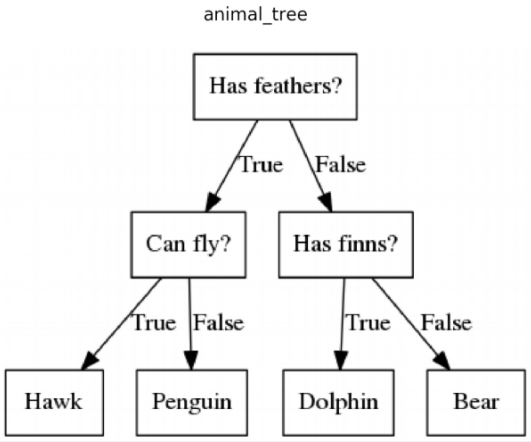
2.4 Random Forests

2.4.1 Technical Explanation

To properly explain the workings of random forests, it helps to first discuss decision tree classifiers. Shortly summarized, a decision tree is a set of rules regarding variable values that continuously splits the data into two groups. All entries satisfying the rule continue to the next rule or result on the left or right, while the entries not satisfying the rule continue to the next rule or result on the opposite side. An example of a rule would be: $x_1 > 10$. This splitting continues until all paths have ended in leaves, which are the final results. Below,

an example of a simple decision tree is given [BS16].

Figure 1: Basic example of a decision tree [tea20]



While it is possible to autonomously implement these rules, there exist functions to do this automatically. Most of these functions work with an entropy value which lies between 0 and 1. When splitting the data into two groups, the diversity of the dependent variable in these entries is analyzed to see the quality of the split. If only one type of outcome exists in a group after a split, then the entropy is 1. The more uneven the distribution is, the closer the entropy is to 1 and the better quality of the split is. This variable allows functions to select the best possible splits so that the most efficient tree can be created. The splitting of a group stops when a high enough entropy is found in a group or the maximum tree depth is reached. The most common value for the dependent variable is selected as the output for that leaf. The maximum tree depth and minimum entropy level are some of the hyper-parameters that are often autonomously selected before creating a tree [BS16].
Once this decision tree is created, the dependent variable of an entry can be predicted by following the decision tree rules down to the bottom, ending up with a result. Decision trees work well for categorical variables both for the dependent and independent input variables. For categorical independent variables, it is easy to compute entropy as a simple count of all discrete value can be done. This is arguably easier than for predicting numerical values, in which case weights come into play. Since the final output of a decision tree can only be a number of discrete leafs, this works well when working with categorical variables [BS16].
Random forest is the expansion of decision trees. For a random forest, a numerous amount of decision trees are trained on randomly selected entries from the data, as well as only being able to split on randomly selected features from these entries. Once all decision trees (the forest) are made, predicting new values of

entries is done by going through each decision tree and using a voting system (possibly with weights) to select the final output. Random Forests are more reliable than decision trees as all variables are looked at. The final structure of a decision tree is highly dependent on the training data, meaning that it is likely to produce results with high variance on new entries. Random Forest reduces this dependency by creating trees with randomly selected training data and selected variables. All the individual trees are overfit, but together they are consistent [BS16].

2.4.2 Properties

To properly predict independent values, Random Forests assume that the output can be found by following a set of rules inferred from the data. While training the trees takes some time, once they are created, getting results does not take long. It is also possible to use multiple processors to make trees at once, since the trees themselves are completely independent of one another [BS16].
Decisions trees in general are easy to understand and explain, as the entire algorithm can be presented and viewed in a single structure. Random forests however, fail to maintain this quality, as understanding many trees at once that are randomly generated is not equally plausible.

2.5 Gradient Boosting

2.5.1 Technical Explanation

Gradient boosting (sometimes called Gradient Boosting Machine) is a machine learning technique that aims to find the best possible fitted prediction function by iteratively learning from predecessor models referred to as weak models. The process of finding this prediction function contains a couple of steps [NK13].
First, a loss function must be selected, which is able to compare prediction values and actual values. There are a lot of possible options to choose from, although the function should be differentiable. Then, a very simple prediction function is used as a baseline. For example, the mean of the dependent variable. Now, the iterative process of the Gradient Boosting method starts. Every cycle starts with letting the old prediction function make predictions. These prediction values are compared to the actual values in the data using the loss function. For every predicted point, the gradient/slope of their position in the loss function is taken. These gradients are then used as the dependent variable to fit the next weak model. Finally, a function is used to decide how much of the new model should be used in conjunction with the old model. Deciding how much a new model is used is simply done with a learning rate value that is multiplied with the new model. This process makes sure that the new result from the loss function will be as small as possible. This process is iterated until a low enough loss is reached, or a predetermined amount of weak models have been made. The final weak model that is fitted is also the final prediction function [NK13].

Which machine learning model function is used in gradient boosting can differ. Both regression models and decision trees could be used, each with their own advantages and disadvantages. This model can clearly work to predict categorical variables as well, since its foundation can be various machine learning algorithms which support categorical variable prediction.

2.5.2 Properties

There is no specific assumption that gradient boosting makes about the context of the data, as this assumption is mostly dependent on the model type that is being used. While methods like Random Forests try to cancel out potential model misspecifications using randomness, Gradient Boosting does so by learning from previous mistakes, which is inherently more data-driven. Even though this, along with the choice of loss function and model type, allows for more experimentation and potentially more accurate models using gradient boosting, its iterative nature makes it prone to overfitting. There are measures to get around this though. Once the function is trained, getting new predictions does not take long, but another issue with the sequential process of gradient boosting is that it can not be sped up using parallel processes, as every iteration relies on the previous one. Gradient boosting also suffers when it comes to transparency. It is not an easy task to distinguish which factors are being taken into account and to what amount after many iterations of optimizing the function. Especially for people without a machine learning background [NK13]. That being said, Gradient Boosting is still a very potent prediction algorithm with the potential for highly accurate results in return for high computation costs.

2.6 Model Decision

2.6.1 Overview

To select the model that will be working alongside logistic regression, first all of the advantages and disadvantages of the discussed models regarding the relevant performance measures are summarized.
K-Nearest Neighbors is a model that is both easy to understand and very transparent, its major drawback however is the computational inefficiency when it comes to large data with many dimensions [FT22] .
Random Forests can be computationally efficient as they can be constructed in parallel, and have shown to deliver consistent results with low variance regarding new predictions. As a result of the large amount of randomly generated decision trees however, most of the transparency and some of the explainability of single decision trees has been lost [BS16].
Finally, Gradient Boosting is a computationally expensive algorithm that shows a lot of promise for accurately fitted prediction functions in return. There is however a risk of this algorithm overfitting, as well as requiring many autonomously

decided hyper-parameters and functions. Transparency and explainability is also lost in the process, as it is hard to keep track of the algorithm's thought process through each iteration [NK13].

2.6.2 Conclusion

For the machine learning model to work alongside logistic regression in the multiply robust imputation method, the Random Forests algorithm was ultimately selected for a multitude of reasons.
Firstly, it is computationally relatively efficient. Once the random forest is constructed, new predictions barely take any time; similarly to logistic regression, where the regression functions have to be constructed.
Transparency is lost when creating a random forest, but there is still an element of explainability, as decision trees are intuitively easy to understand [BS16].
There is also an argument to be made about the distinct nature of Random Forests' assumption about the context of the data. Logistic regression assumes that the correct value of the dependent variable can be reached with a formula, where independent variables are given weights. Random Forests assume that the correct value of the dependent variable can be reached by following a set of rules derived from the data. These different perspectives on the context can give the multiply robust imputation method more variety and therefore a higher chance of containing a correctly specified model.
There are also other reasons as to why the Random Forests algorithm was selected instead of KNN or Gradient Boosting. While the transparency and explainability of Random Forests are not as good as KNN, the computational efficiency makes up for it by being more compatible with the computational costs of logistic regression. Gradient Boosting shares many of the same qualities as the Random Forest model, however Gradient Boosting has many different hyper-parameters and function selections to choose from. This might make it more difficult to reason about the final experiment results, as it might be hard to pin-point how much of an effect these choices have made.

3 Experiment

3.1 Section overview

This section will be divided up into three subsections. First, the mean idea behind the experiment and the process is explained. After this, the setup of the experiment is covered. This subsection will cover details such as the working environment, coding and the origins of used data. The fourth and final subsection will cover the results of the experiment.

3.2 Experiment Explanation

For the current study, an experiment will be done to analyze the change in performance of a Multiply robust imputation method when using either only logistic regression models or a combination of logistic regression and random forest models. The main idea behind the experiment is that using multiple types of models together might influence the performance of the complete model by means of working together or possibly against each other. Reasons as to why these specific models were chosen are given in the model selection section of this thesis.
The imputations will be performed on multiple data sets of different natures that all have a single specified variable that might be missing, as opposed to multiple variables having the chance of missing values for a single entry. As for comparing performance, analyses will be made regarding the models prediction accuracies, confusion matrices and estimator bias.

3.3 Experiment Set-up

3.3.1 Working Environment and Coding Language

The creation of the model, along with the importing and pre-processing of the data has all been done with Python using Jupyter Notebook, which is a web-application that allows the user to execute Python code in cells/blocks at a time. This feature made it an appealing choice for this experiment, as it allows for quick changes to small parts of the code which can be executed separately instead of having the need to execute the complete code for every minor change. Python was chosen as the coding language because of personal experience and its nature that allows it to work efficiently with data sets.

3.3.2 Coding and Algorithm

This subsection will shortly cover the workings of the code/algorithm I made to execute the experiment.
First, all coding libraries that will be used are imported. These include NumPy for mathematics, Pandas for data frame operations, Seaborn for plots, Random for sampling and finally Scikit-Learn for the logistic regression and random forest algorithms. Then, the data is imported and pre-processed for later use.

The bulk of the code is a function called 'Multiply Robust Imputation Experiment' (visible in the appendix). As input, this function receives the data, along with the amount of models it should make, the sample size that is used for training and testing each of these models, the list of predictor variables and the predicted variable and finally whether the imputation method should only use logistic regression models, or a combination of logistic regression and random forest models.
As output, the algorithm returns the accuracy of the final predictions, a confusion matrix for the final predictions and a dictionary containing the estimated

bias for each of the possible values from the predicted variable.

A short pseudocode of the imputation algorithm for clarity:

-Sample size is computed based on fraction and the size of the input data
-For each model that needs to be created:
—Create a list of lists of variables that are a random subset of all independent variables
—Create a list of random samples with predetermined sample size of the full data set
—Split this random sample into a training and test sample
-Merge all test sets from the previous samples to create the final test set on which predictions are made
-Each list of variables and training sample is used to train one machine learning model (or two in case mixed models is enabled)
-Predictions are made using each model
-Final predictions are decided using a majority vote among all models

Important to note is that the algorithm always favors logistic regression when it comes to an even amount of models. Furthermore, when a mixed set of models is created, the algorithms makes sure that all of the random forest models are trained on the same data samples and variable samples as the logistic regression model.

3.3.3 Data

For the experiment, three data sets have been used. They have been selected for satisfying a couple of personally chosen criteria. They had to have more than 500 observations, so that the results would have a smaller chance of being largely variant. Secondly there had to be between 5 and 20 predictor variables present, so that the generated models are not too simple or complicated. Finally, the predicted variable that depends on the previously mentioned predictor variables also needed to be multivariate in nature with not too many possible values (between 3 and 10).
The first data set that was selected is called 'Airbag and other influences on accident fatalities' and features information about car accidents. Factors like whether the airbags were deployed and the driver's age are taken as independent variables and the severity of the injuries resulting from the crash as the dependent variable with 5 possible options. Most of the predictor variables are binary in nature.
The second data set that was selected is called 'Number of Equations and Citations for Evolutionary Biology Publications' and features data on papers' citations in journals. The properties of these papers as well as their citations are used as independent variables. In which journal they were originally posted is used as the dependent variable with three possible options. Most of the predictor variables are numerical in nature.

The last data set that was selected is called 'Deidentified Results of COVID-19 testing at the Children's Hospital of Pennsylvania (CHOP) in 2020' and features data about COVID-19 tests in 2020 from a hospital in Pennsylvania. Factors such as the result of the test, the test subjects' age and how far along into the pandemic the test was are used as independent variables. Which group of people paid for the test (government, commercial, self pay etc.) is the dependent variable. The predictor variables are both binary and numerical in nature.
Sources for these data sets are in the appendix.

3.4 Results

To eliminate the chance of using improper values for the amount of models or sample size, a wide range of possible values is covered for each of the data sets. Regarding the amount of models generated, the values are 2, 5, 10, 20 and 30. Regarding the sample size, this has been done in the fractions (in comparison to the full data) 1/2, 1/5, 1/10, 1/20, 1/30. For each data set, every single combination of these two ranges is run 40 times. 20 times for a non-mixed set of models and 20 times for a mixed set of models. The results that are talked about hereafter are the averages of these 20 runs. An individual logistic regression model and random forest model is also used to make predictions on the data as a baseline to compare to. These individual predictions were also run 20 times on different data. Tables of the exact results are visible in the appendix.
For analysis of the results, firstly accuracy will be covered, then the confusion matrices and finally the estimator bias.

3.4.1 Accuracy

The accuracy of the model is obtained by dividing the amount of correct predictions that the imputation method makes by the amount of total predictions, as shown below.

$$Accuracy = \frac{CorrectPredictions}{TotalPredictions}$$

When it comes to accuracy, discussing the exact numbers is not very useful here, as they differ across the data sets for obvious reasons. Only comparisons will be dealt with instead.
Individually, there is a healthy balance between the accuracy of logistic regression and random forests on all three data sets. Logistic regression outperforms random forest in the first dataset by a bit, however it is outperformed by random forests by the same margin in the other two sets. This is nice to know, as it indicates that the results cannot be blamed on the nature of the models that are used alone.
For the multiply robust imputation, there does not seem to be a great effect on the accuracy by tweaking the amount of models trained or the sample sizes

used. Small sample sizes often lead to more variance in the accuracy however, while for large sample sizes the opposite occurs.
The accuracy of the multiply robust imputation method that only contains logistic regression models is lower than the method that contains a mix of models for all three data sets. For the first data set, this difference is 1.2 percent. For the second and third data set this difference is only 0.3 percent and 0.5 percent respectively.
Finally, it is interesting to note that not one of the multiply robust imputation methods got an accuracy score that was much higher than the individual models.

3.4.2 Confusion Matrix

For each time that the imputation method is run, a confusion matrix is obtained. This matrix shows the predicted values on the Y-axis and the actual values on the X-axis. Each cell in the matrix contains the number of entries that were correctly or incorrectly classified.

Figure 2: Example of a confusion matrix from a single run of the algorithm on the airbag data

One thing that all three data sets have in common is that their dependent variable has one or two possible values that occur much more often than others. This dominance in appearance is increased by the machine learning models and is therefore also visible in the confusion matrices produced by the multiply robust imputation predictions. Pretty much all of the predictions made by the imputation method fall into a dominant category. For the imputation with the

mixed models, this dominance is less extreme.
In general, if the imputation contains more models with a big sample, the dominance is increased, while having many models with a small sample decreases this dominance and allows for more variety in predictions. This is likely due to the increase in variance from the small sample size. If only a few models are trained, the dominance is always increased. These properties do not change for the imputation method with the mixed models.

3.4.3 Estimator Bias

Besides looking at correct predictions, an imputation method's performance can also be evaluated by looking at the value distribution before imputation and after imputation. For each possible value of the dependent variable, their proportion can be compared from before and after the imputation to obtain their respective bias estimator, which signifies how much more or less often they occur in the data compared to before the imputation. A simple formula is given below, where θ is the actual value proportion and $\hat{\theta}$ is the predicted value proportion.

$$Bias(\hat{\theta}) = \hat{\theta} - \theta$$

An ideal imputation method would not have any bias.

As recording the estimator bias for every value for every run would be a vast amount of work, only the biggest estimator bias is recorded per run. This bias estimator always belongs to the most dominant category, which makes sense, as its large proportion size will be represented the most while fitting the machine learning models.
After recording the highest estimator bias for 20 runs for each combination of number of models and sample size (similarly to the accuracy), there does not appear to be any difference between the average largest estimator bias of the imputation method without mixed models and the imputation method with mixed models. The differences are all very small.

4 Conclusion

To answer the research question: 'Which performance-based differences are there between a multiply robust imputation algorithm that uses only logistic regression models, in comparison to a multiply robust imputation algorithm that uses another type of model alongside logistic regression?', three important pieces of information have been gathered from the data experiment to aid in giving an answer to this question.
The first of which is that the accuracy of the imputation method with the combined set of models is actually consistently higher than the accuracy of the imputation method using only logistic regression models. It might be a bit of

a stretch to call this conclusive evidence for calling an imputation method with a combined set of models the best method to use in all cases. Not only is this information only based on a handful of data sets (although they are different in nature), the increases in accuracy range from small to minor. Then there is also the argument that all three data sets had very dominant categories, which might have made it easier to accidentally get the answer right. Since most models had around a 40-50 percent accuracy, this might not apply though. That being said, as random forest is model that has a significantly different outlook on the context of the data than logistic regression. The increase in accuracy is a clear sign that in this case, the extra perspective has allowed for better prediction accuracy.
Additionally, this increase in accuracy gives us a compelling argument that the models are not actively working against each other. One might argue that only having half the amount models from two types instead of a larger amount of models from a single model type could result in neither of the models living up to their potential, resulting in an overall lower accuracy. This issue certainly does not occur according to our experiment results. Random forest was mostly selected as the model to use in the multiply robust imputation method as its assumptions on the data are vastly different from those that logistic regression uses. Although it would be tough to conclude from this experiment alone, it does appear as though the additional model type's prediction methods allows the primary model type to get more imputations correct using the extra insight.
The second piece of information we have is that the largest estimator bias from the imputation method with the combined set of models does not increase drastically compared to the imputation method with only logistic regression. This does not show that the combined models remove previous bias, it shows more so that they do not create additional bias that was not already in the data itself.
Finally, it might be noteworthy to mention that the diversity of predictions increases when the logistic regression model and random forest models are used in tandem, even though the accuracy stays relatively the same. This is likely a result of the models having different assumptions of the context of the data. While these assumptions might not be as different for any combination of models as they are here, prediction diversity could be an interesting or useful addition to imputation or classification machine learning algorithms.

These results together bring us close to an answer to our research question. It seems like using a combination of two types of models that differ in outlook on the data does seem to give extra insight for proper missing data imputation, at least for logistic regression and random forests. This comes in the form of a bigger chance of proper model specification and an increase in predictions diversity. Furthermore, no additional estimator bias is created.

5 Discussion

The main problem for this thesis is that it is a near impossible task to find a conclusive answer to the research question that was selected. This is because there are an innumerable amount of model combinations out there all with their own assumptions about the context of data. Instead of finding a conclusive answer, an approach was taken to show an example of two models working together in pursuit of gaining more knowledge about the topic of combined machine learning models. The models that were selected in this thesis do provide a good example of how multiple models can be used together to function as something greater than the sum of their parts. The selection for these models is also backed up with research, which allows the results to be interpreted into the larger picture. While this might be a way of contributing to the knowledge about this topic, with more time, it could have been explored a bit further. Regarding further research, instead of settling with two models, multiple models could have been used. All possible combinations could be examined, which might give a less specific picture of the fundamentals of models working together than is visible in this thesis.

Then there is also the potential issue with the lack of data sets. While regarding their independent variable types they are different in nature, they all share the same type of value distributions regarding the dependent variable, with one or two values occurring much more often than the others. Running the same experiment on a data set with an approximately equal proportion for each value of the dependent variable might yield new results.
Finally, there is the potential issue of the significance of the results as discussed in the conclusion. The increase in accuracy is relatively minor. As was discussed in the conclusion though, learning that the models are not actively working against each other is something that can be conclusively established. This is more important for this thesis as the main goal was to find out more about the cooperation between different types of models.

6 Appendix

Data source url:
https://vincentarelbundock.github.io/Rdatasets/datasets.html
Data set names:
Airbag and other influences on accident fatalities
Number of Equations and Citations for Evolutionary Biology Publications
Deidentified Results of COVID-19 testing at the Children's Hospital of Pennsylvania (CHOP) in 2020

Data Experiment Results:

Experiment model Airbag results (avg of 20 runs)

Accuracy Not mixed (Avg:36.924)

SampleFraction/Models	2	5	10	20	30
1/2	40.9%	40.8%	39.2%	35.5%	32.1%
1/5	42.1	41.8	36.0%	34.2%	33.1%
1/10	42.6%	40.7%	38.5%	35.1%	30.9%
1/20	42.1%	40.9%	37.1%	31.1%	30.5%
1/30	38.9%	40.6%	35.4%	33.8%	29.2%

Highest proportion bias (Avg: 0.3496)

SampleFraction/Models	2	5	10	20	30
1/2	0.542	0.31	0.384	0.362	0.414
1/5	0.306	0.328	0.390	0.363	0.364
1/10	0.211	0.443	0.416	0.396	0.413
1/20	0.256	0.353	0.353	0.341	0.320
1/30	0.292	0.392	0.358	0.384	0.328

Accuracy Mixed (Avg: 37.88)

SampleFraction/Models	2	5	10	20	30
1/2	42.5%	43.3%	39.9%	35.5%	36.0%
1/5	40.5%	41.9%	39.5%	35.8%	33.5%
1/10	39.5%	38.4%	39.5%	35.2%	34.0%
1/20	40.5%	43.3%	39.9%	33.5%	32.2%
1/30	40.2%	40.9%	38.8%	32.8%	29.9%

Highest proportion bias (Avg: 0.37104)

SampleFraction/Models	2	5	10	20	30
1/2	0.549	0.301	0.473	0.333	0.335
1/5	0.448	0.411	0.287	0.264	0.340
1/10	0.331	0.359	0.373	0.405	0.417
1/20	0.486	0.223	0.298	0.351	0.299
1/30	0.54	0.47	0.28	0.348	0.355

Individual accuracies:

LogReg: 43.4%

RandomForest: 40.3%

Experiment model Journal results (avg of 20 runs)
Accuracy Not mixed (Avg: 53.14)

SampleFraction/Models	2	5	10	20	30
1/2	0.639	0.534	0.542	0.528	0.530
1/5	0.618	0.524	0.541	0.530	0.526
1/10	0.548	0.529	0.535	0.537	0.514
1/20	0.532	0.5	0.497	0.505	0.534
1/30	0.5	0.511	0.491	0.536	0.503

Highest proportion bias (Avg: 0.18308)

SampleFraction/Models	2	5	10	20	30
1/2	0.416	0.255	0.289	0.236	0.268
1/5	0.242	0.246	0.167	0.181	0.158
1/10	0.121	0.116	0.322	0.157	0.132
1/20	0.065	0.073	0.118	0.128	0.120
1/30	0.277	0.120	0.182	0.126	0.062

Accuracy Mixed (Avg: 52.76)

SampleFraction/Models	2	5	10	20	30
1/2	0.698	0.574	0.551	0.526	0.528
1/5	0.6	0.546	0.507	0.532	0.543
1/10	0.533	0.543	0.527	0.523	0.536
1/20	0.486	0.544	0.605	0.483	0.473
1/30	0.424	0.520	0.437	0.417	0.535

Highest proportion bias (Avg: 0.19656)

SampleFraction/Models	2	5	10	20	30
1/2	0.473	0.314	0.243	0.202	0.281
1/5	0.203	0.315	0.312	0.228	0.203
1/10	0.073	0.108	0.106	0.195	0.163
1/20	0.231	0.189	0.073	0.085	0.122
1/30	0.314	0.102	0.077	0.099	0.203

Individual accuracies:
LogReg: 63.8%
RandomForest: 67.7%

Experiment model Covid results (avg of 20 runs)
Accuracy Not mixed (Avg: 44.49)

SampleFraction/Models	2	5	10	20	30
1/2	0.453	0.454	0.449	0.439	0.443
1/5	0.444	0.450	0.453	0.446	0.443
1/10	0.459	0.455	0.444	0.447	0.443
1/20	0.445	0.438	0.448	0.443	0.441
1/30	0.426	0.435	0.456	0.435	0.434

Highest proportion bias (Avg: 0.16088)

SampleFraction/Models	2	5	10	20	30
1/2	0.167	0.339	0.171	0.234	0.101
1/5	0.116	0.185	0.207	0.121	0.128
1/10	0.284	0.133	0.086	0.188	0.068
1/20	0.262	0.089	0133	0.074	0.119
1/30	0.197	0.268	0.131	0.067	0.154

Accuracy Mixed (Avg: 44.7)

SampleFraction/Models	2	5	10	20	30
1/2	0.448	0.457	0.453	0.447	0.446
1/5	0.474	0.459	0.457	0.448	0.445
1/10	0.449	0.451	0.452	0.442	0.447
1/20	0.467	0.440	0.445	0.440	0.421
1/30	0.465	0.427	0.434	0.442	0.420

Highest proportion bias (Avg: 0.1488)

SampleFraction/Models	2	5	10	20	30
1/2	0.230	0.130	0.268	0.0189	0.106
1/5	0.184	0.121	0.174	0.209	0.128
1/10	0.154	0.325	0.068	0.132	0.222
1/20	0.075	0.099	0.268	0.085	0.065
1/30	0.201	0.085	0.081	0.114	0.077

Individual accuracies:
LogReg: 47.0 %
RandomForest: 50.2%

Below follows the code for the multiply robust imputation algorithm that was used for the experiment of this study:

```
#Multiply robust imputation function with mixed model option
def Multiply_Robust_Imputation_Experiment(data, X, Y, nr_models, fraction, mixed):


    #Splitting the amount of used models in two for mixed imputation
    #and returning an error if a number is requested that is too small
    if nr_models <= 1:
        return "Number of models must be greater than 1"

    log_reg_models = nr_models

    if mixed == True:
        log_reg_models = round(int((nr_models) / 2))+1
        forest_models = nr_models-log_reg_models

    #Training all the used models and adding them to a list
    variable_samples_list = []
    training_sets = []
    test_sets = []

    #Getting the size for each sample using fraction
    sample_size = round(len(data)*fraction)

    #Making all of the samples and variable-lists to be used
    for i in range (log_reg_models):
        variable_sample = random.sample(X, random.randint(1,len(X)))
        variable_samples_list.append(variable_sample)
        sample = data.sample(n=sample_size)
        training_sample = sample.sample(frac=0.9)
        test_sample= sample.drop(training_sample.index)
        training_sets.append(training_sample)
        test_sets.append(test_sample)

    #Combining all the test sets for the final predictions
    Combined_Test_Data = test_sets[0]
    for i in range (len(test_sets)):
        Combined_Test_Data = pd.concat([Combined_Test_Data, test_sets[i]])

    #Making a dataframe to put all the predictions into
    All_Predictions = pd.DataFrame()
    All_Predictions['Actual_Values'] = Combined_Test_Data[Y]

    for i in range (log_reg_models):
        All_Predictions["LogReg_Model" + str(i+1)] = LogisticRegression().fit(training_sets[i][variable_samples_list[i]], traini

    #Making additional Random Forest Models if these are requested
    if mixed == True:
        for i in range(forest_models):
            All_Predictions["Forest_Model" + str(i+1)] = RandomForestClassifier(max_depth = 4).fit(training_sets[i][variable_sam


    All_Predictions = All_Predictions.reset_index()
    All_Predictions = All_Predictions.drop(columns=['index'])

    #Deciding the final predictions using majority vote
    column_names_list = All_Predictions.columns.values.tolist()
    final_predictions = []

    #For each row, create a dictionary and choose the most common
    for index, row in All_Predictions.iterrows():
        votes_dictionary = {}
```

```
    for column_name in range (2,len(column_names_list)):

        votes_dictionary[row[column_name]] = votes_dictionary.get(row[column_name], 0) + 1
    final_predictions.append(max(votes_dictionary))

All_Predictions.insert(1, 'Final_Predictions', final_predictions)

#Returning performance measurements
actual_values = All_Predictions['Actual_Values'].to_numpy()
final_predictions = All_Predictions['Final_Predictions'].to_numpy()

accuracy = accuracy_score(actual_values, final_predictions)

results_confusion_matrix = confusion_matrix(actual_values, final_predictions)

df_cm = pd.DataFrame(results_confusion_matrix, index = [i for i in ["Incapacity", "Killed", "No Incapacity", "None", "Possib
            columns = [i for i in ["Incapacity", "Killed", "No Incapacity", "None", "Possible Injury"]])
plt.figure(figsize = (10,7))
sn.heatmap(df_cm, annot=True)

#Proportion calculations
proportions = (data['injSeverity'].value_counts()).to_dict()
proportions = {i: round(j/len(data), 3) for i, j in proportions.items()}
proportions = {i: j for i, j in sorted(proportions.items())}

#Making a list containing all the prediction proportions to calculate the bias of estimated category proportions
counts_dict_list = []

for column in All_Predictions.columns:
    if column == 'Actual_Values' or column == 'Final_Predictions':
        continue
    counts_dict = (All_Predictions[column].value_counts()).to_dict()
    counts_dict = {i: round(j/len(All_Predictions), 3) for i, j in counts_dict.items()}
    counts_dict = {i: j for i, j in sorted(counts_dict.items())}
    counts_dict_list.append(counts_dict)

Full_Dict = counts_dict_list[0]
for i in range(1,len(counts_dict_list)):
    for key in set(Full_Dict) | set(counts_dict_list[i]):
        Full_Dict[key] = Full_Dict.get(key, 0) + counts_dict_list[i].get(key, 0)

Full_Dict = {i: round(j/len(counts_dict_list),3) for i, j in Full_Dict.items()}

Bias_Dict = {}
for key in set(proportions) | set(Full_Dict):
    Bias_Dict[key] = round(Full_Dict.get(key, 0) - proportions.get(key, 0),3)


#Return the accuracy, final predictions and estimator biases
return accuracy, All_Predictions, Bias_Dict
```

References

[Alk12] Hussain Alkharusi. Categorical variables in regression analysis: A comparison of dummy and effect coding. *International Journal of Education*, 4(2):202, 2012.

[BS16] Gérard Biau and Erwan Scornet. A random forest guided tour. *Test*, 25:197–227, 2016.

[BSGB22] Michael Bücker, Gero Szepannek, Alicja Gosiewska, and Przemyslaw Biecek. Trans-

parency, auditability, and explainability of machine learning models in credit scoring. *Journal of the Operational Research Society*, 73(1):70–90, 2022.

[CHS21] Sixia Chen, David Haziza, and Alexander Stubblefield. A note on multiply robust predictive mean matching imputation with complex survey data. *Survey Methodology*, 47(1):215–223, 2021.

[FT22] Shahla Faisal and Gerhard Tutz. Nearest neighbor imputation for categorical data by weighting of attributes. *Information Sciences*, 592:306–319, 2022.

[Gui14] Montserrat Guillén. Regression with categorical dependent variables. *Predictive modeling applications in actuarial science*, 1:65–86, 2014.

[Li15] Stuart E. A. Allison D. B. Li, P. Multiple imputation: A flexible tool for handling missing data. 2015.

[NK13] Alexey Natekin and Alois Knoll. Gradient boosting machines, a tutorial. *Frontiers in neurorobotics*, 7:21, 2013.

[SAPV19] Catia M Salgado, Carlos Azevedo, Hugo Proenca, and Susana M Vieira. Missing data. 2019.

[tea20] Editorial team. Decision trees explained with a practical example.

https://towardsai.net/p/programming/decision-trees-explained-with-a-practical-example-fe47872d3b53, 2020.

[TL10] Scott Tonidandel and James M LeBreton. Determining the relative importance of predictors in logistic regression: an extension of relative weight analysis. *Organizational Research Methods*, 13(4):767–781, 2010.

[ZHYH17] Muhan Zhou, Yulei He, Mandi Yu, and Chiu-Hsieh Hsu. A nonparametric multiple imputation approach for missing categorical data. *BMC medical research methodology*, 17(1):1–12, 2017.