Deep Imputation Networks vs. Mean Imputation in Tabular Learning with Missing Data

Tomas Miskov

Masters thesis for
MPhil Business Datascience

May 16, 2024

Abstract

In the context of supervised learning, missing values pose a serious problem as most supervised learning algorithms cannot inherently deal with missing data. There exists a rich literature in missing value imputation for inference - the task of imputing missing values as accurately as possible. But only recently, theoretical results for imputing missing values such that the performance on the downstream supervised learning task is maximized have been established. These results suggest that there are benefits in developing new supervised learning algorithms that jointly learn the imputation and the prediction function. This should in theory ensure that both functions are smooth, continuous and thus easily learnable on a finite sample of training points. We put these assertions to test in a computational study with 10 real tabular datasets where we compare two deep neural networks that handle missing data natively against impute-then-regress procedures that first impute the missing values and then train a predictive model on the imputed dataset. One of the tested neural networks, dubbed the *Dropout-like* network, is our original contribution to the sub-group of neural networks that can be trained on datasets with missing values. We find that the recent theoretical results do not hold up in practice and mean imputation followed by training a predictive model is as performant as the newly developed deep imputation networks, including the *Dropout-like* network. This suggests that practitioners are better off sticking to simpler methods of mean imputation rather than investing time and resources into more complex deep imputation models.

Keywords: missing value imputation, mean imputation, impute-then-regress, deep imputation networks

Contents

1 Introduction

Missing values naturally occur in datasets for multiple reasons. Whether it is data entry errors, privacy protection or simple data loss, researchers and practitioners must first handle missing data before they can carry on with inference or prediction. For inference, a fundamental principle adopted to address missing values is the Missing At Random (MAR) assumption. This principle suggests that the absence of data in certain features does not depend on the missing data itself but rather on the values of the features that are observed (Rubin, 1976). The values of the observed feature can thus be used to infer the missing values, which is the idea employed by methods like expectation maximization (Dempster et al., 1977) or in multiple imputation strategies like MICE (Azur et al., 2011).

When it comes to prediction - a distinct task from inference - the strategies diverge. While there are some prediction models, particularly those based on decision trees, that can naturally accommodate missing data, the easiest strategy is to drop the data points with missing values and fit a predictive model on the remaining observations. This is only possible when a small fraction of data points contains missing values otherwise too little data remains for model fitting. However, even if deletion does not result in a substantial data loss, the process may shift the underlying data distribution and introduce unwanted bias. Thus, the alternative, in which two main methodologies compete, is missing value imputation. In the imputation paradigm, the prevalent approach mirrors that of inference: impute first, predict second. This impute-then-predict, also known as impute-then-regress strategy, relies heavily on the MAR assumption to ensure accurate imputations. Under this paradigm the missing values are first imputed and then a predictive model is fitted on the imputed dataset. The competing approach is to develop predictive models that at the same time learn the optimal imputation function and the optimal prediction function. Recent developments have shown that models, especially neural networks, can be designed in this way, handling missing values natively, without the need for a separate imputation stage (Ipsen et al., 2022; Le Morvan et al., 2021; Śmieja et al., 2018).

These models challenge the earlier approaches in that they claim to lead to better predictive performance when optimization of imputation and prediction is performed jointly. The claims are theoretically supported by recent results in the asymptotic regime, when the size of the training sample grows to infinity (Le Morvan et al., 2021). However, the empirical evidence on real, finite datasets is still scarce (Sun et al., 2023; Van Ness and Udell, 2023). This thesis thus aims to contribute to the discussion on the impact of missing values on prediction and the evolving methods to address them. By examining both traditional impute-then-regress frameworks and deep-imputation models, we aim to bridge theoretical insights with practical challenges, clearing the path forward for predictive modeling in the presence of missing data. Our investigation contrasts the recent theoretical results with empirical validation and provides a critical look at the emerging deep-imputation models and their predictive performance as compared to more traditional impute-then-regress approaches. Our empirical validation focuses on tabular datasets that come from different domains, be it medical, financial, textual or data from physical experiments. We compare deep-imputation models with impute-then-regress approaches that either use neural networks, for a fair comparison, or random forests, as a powerful alternative to neural networks, especially for tabular learning (Grinsztajn et al., 2022).The question we aim to answer is: *Do the theoretical results of deep-imputation models translate to practice or are practitioners better off using traditional impute-then-regress approaches in the setting of supervised learning with missing data?* Our analysis and findings provide three key contributions:

(i) We introduce a new deep-imputation model, dubbed the *Dropout-like* network, whose simplicity results in faster training times while still achieving equivalent or better performance on the downstream prediction task than the existing deep-imputation models.

(ii) We contribute to the existing body of empirical results by comparing impute-then-regress methods with deep-imputation models on 10 real-world datasets with both synthetic and *real* missingness patterns.

(iii) We show that although these deep-imputation models have theoretical grounding, their performance on real datasets merely matches the performance of simple strategies like mean imputation, suggesting the simple approach should be favored in practice.

The rest of this thesis is organized as follows. In Section 2, we establish the problem setting and discuss the underlying theory and practical results for supervised learning with missing data. Section 3 then introduces two deep imputation neural networks that can natively handle missing data and learn the imputation and the prediction function jointly. We put these networks to test against impute-then-regress approaches in Section 4, where we discuss our experimental setup, results of our experiments, and their implications. The discussion of these results is then offered in Section 5, after which concluding remarks and directions for future work are addressed in Section 6.

2 Related Work

When imputing missing values there are two general metrics one can care about - *imputation accuracy* and *downstream predictive performance* (*e.g.*, classification accuracy or mean squared error in regression). Imputation accuracy measures how accurate our model is at imputing the missing values. This is generally the metric of interest in the inference task where the aim is to infer the true unobserved values of the missing variables based on the observed variables (the MAR assumption). Accurate inference is useful in *e.g.*, unsupervised learning because it helps uncover the underlying structure of the data without relying on labeled outcomes. However, imputation accuracy can only be measured in datasets with synthetically masked entries where we have access to the ground truth. This ground truth is never available in practice, thus imputation accuracy can never be directly measured in datasets with real missingness. On the other hand, the metric of downstream predictive performance is concerned with generating imputations that maximize the performance of the downstream predictor - be it in a regression or a classification problem. This thesis is concerned with the latter metric, namely what is the best practical approach for imputation of missing values when the goal is not necessarily accurate inference but rather improved predictive power? We thus define the problem setting and discuss the related results for the prediction task, touching on the inference rather briefly at places where it establishes the basis for developing imputation models for the predictive setting.

2.1 Problem Setting

We consider the problem of predicting an output variable y from a set of features, encapsulated in a vector $\mathbf{x}$, in the presence of missing values. Specifically, we denote $\mathbf{x} \in \mathbb{R}^d$ as the complete feature vector, $y \in \mathbb{R}$ as the response variable, and $\mathbf{m} \in \{0,1\}^d$ as the missingness indicator vector, where each entry m_j for $j \in \{1, \ldots, d\}$ indicates whether the corresponding feature x_j is missing ($m_j = 1$) or is observed ($m_j = 0$). The dataset consists of n independent and identically distributed (i.i.d.) samples $\{(\mathbf{x}_i, \mathbf{m}_i, y_i)\}_{i=1}^n$, where $\mathbf{x}_i$ represents the feature vector, $\mathbf{m}_i$ is the missingness pattern vector indicating which values in $\mathbf{x}_i$ are missing, and y_i is the observed response variable for the i-th sample.

For a given sample $(\mathbf{x}_i, \mathbf{m}_i, y_i)$, we define the set of indices for missing and observed values as $mis(\mathbf{m}_i) = \{j | m_{ij} = 1\}$ and $obs(\mathbf{m}_i) = \{j | m_{ij} = 0\}$, respectively. The observed portion of the feature vector is then denoted by $\mathbf{x}_{obs(\mathbf{m}_i)}$, comprising only the entries of $\mathbf{x}_i$ that are observed (*i.e.*, for which $m_{ij} = 0$). Conversely, $\mathbf{x}_{mis(\mathbf{m}_i)}$ are the entries corresponding to missing values. To lighten the notation, we will refer to these two subsets of the feature vector $\mathbf{x}$ as $\mathbf{x}_{i,obs}$ and $\mathbf{x}_{i,mis}$. The missing values in $\mathbf{x}_{mis}$ are represented as `NaN`, leading to the incomplete feature vector $\tilde{\mathbf{x}}_i \in (\mathbb{R} \cup \{\texttt{NaN}\})^d$, where $\tilde{x}_{ij} = x_{ij}$ if $m_{ij} = 0$, and $\tilde{x}_{ij} = \texttt{NaN}$ if $m_{ij} = 1$. Finally, we denote the imputed vector of features as $\hat{\mathbf{x}}_i$ where previously missing entries $\mathbf{x}_{i,mis}$ are replaced with some imputed values that are either learned or selected by the researcher (*e.g.*, mean or mode imputation). Table 1 provides a quick lookup summary of the introduced notation.

In addition to the above notation, we will also refer to the dataset $\{\mathbf{x}_i, \mathbf{m}_i, y_i)\}_{i=1}^n$ in its matrix form as $\{\mathbf{X}, \mathbf{M}, \mathbf{y}\}$, where $\mathbf{X}$ and $\mathbf{M}$ are matrices of shape $n \times d$ and $\mathbf{y}$ is a d-long vector of targets. The same matrix notation will be used for the incomplete dataset $\tilde{\mathbf{X}}$ and imputed dataset $\hat{\mathbf{X}}$. To distinguish the targets from their predicted values, we will denote the predictions with a hat notation as $\hat{\mathbf{y}}$. Finally, to succinctly refer to the entire training dataset $\{\mathbf{X}, \mathbf{M}, \mathbf{y}\}$, we adopt the notation of $\mathcal{D}_{\text{train}}$. The same holds for the test dataset $\mathcal{D}_{\text{test}}$.

Notation	**Meaning**
$\mathbf{x} \in \mathbb{R}^d$	Complete feature vector (uknown)
$y \in \mathbb{R}$	Complete response variable (observed)
$\tilde{\mathbf{x}} \in (\mathbb{R} \cup \{\texttt{NaN}\})^d$	Incomplete feature vector (observed)
$\mathbf{m} \in \{0,1\}^d$	Missingness indicator vector ($m_i = 1$ if $\tilde{x}_i = \texttt{NaN}$)
$mis(\mathbf{m})$	Indices of missing values in $\tilde{\mathbf{x}}$
$obs(\mathbf{m})$	Indices of observed values in $\tilde{\mathbf{x}}$
$\mathbf{x}_{obs}$	Observed entries of the feature vector $\tilde{\mathbf{x}}$
$\mathbf{x}_{mis}$	Missing entries of the feature vector $\tilde{\mathbf{x}}$
$\hat{\mathbf{x}} \in \mathbb{R}^d$	Imputed feature vector

Table 1: Summary of the notation

2.2 Bayes Optimal Predictions

In practice, we only observe the incomplete dataset $\{\tilde{\mathbf{X}}, \mathbf{M}, \mathbf{y}\}$ for which we try to learn a predictor $f : \mathbf{X} \to \hat{\mathbf{y}}$ that either takes as input the incomplete feature matrix $\tilde{\mathbf{X}}$ or the imputed feature matrix $\hat{\mathbf{X}}$ and outputs the vector of predictions $\hat{\mathbf{y}}$. To study the optimality of the learned prediction function f, we introduce the concepts of *Bayes consistency* and *Bayes optimality.*

The process of learning the function f involves defining a loss function $\ell : \mathbb{R}^n \times \mathbb{R}^n \to [0, \infty)$ that takes as input the vector of predicted values $f(\mathbf{X}) = \hat{\mathbf{y}} \in \mathbb{R}^n$ and the vector of target values $\mathbf{y} \in \mathbb{R}^n$ and outputs a scalar value that captures how well the predictions match the targets. Learning f involves minimizing the expected value $\mathbb{E}[\ell(f(\mathbf{X}), \mathbf{y})]$ of this loss function. Given this general setting, the best possible prediction function f^* is said to be *Bayes optimal* (Lehmann and Casella, 1998), if it minimizes the expected loss:

$$f^* = \underset{f:\mathbf{X}\to\hat{\mathbf{y}}}{\operatorname{argmin}}\ \mathbb{E}[\ell(f(\mathbf{X}), \mathbf{y})].$$

Thus f^* is the prediction function that outputs the predictions $\hat{\mathbf{y}}$ that are the closest, as measured by ℓ, to the targets $\mathbf{y}$ out of all possible prediction functions. However, the optimality of any prediction function f depends on both the selected learning algorithm and the data points available for training. Learning algorithms vary from gradient-based methods, such as those used in neural networks, to gradient-free approaches, like those for optimizing decision trees and support vector machines. Because of these two factors, we introduce the notation $\hat{f}_{\text{train}}$ to signify that the prediction function is *learned*, conditional on the training procedure, and its performance depends on the finite training sample $\mathcal{D}_{\text{train}}$. With this finite training dataset $\mathcal{D}_{\text{train}}$, we approximate the expected loss through its empirical counterpart: $\sum_{i=1}^{n} \ell(\hat{f}_{\text{train}}(\mathbf{x}_i), \mathbf{y}_i)$. Minimizing this empirical loss yields the conditionally optimal prediction function $\hat{f}^*_{\text{train}}$. If we then let the size of $\mathcal{D}_{\text{train}}$ grow to infinity, $\hat{f}^*_{\text{train}}$ is said to be *Bayes consistent* (Devroye et al., 1996), if it in the limit achieves the same expected loss as the Bayes optimal function f^*:

$$\lim_{n \to \infty} \sum_{i=1}^{n} \ell(\hat{f}^*_{\text{train}}(\mathbf{x}_i), \mathbf{y}_i) = \mathbb{E}[\ell(f^*(\mathbf{X}), \mathbf{y})].$$

For a prediction function to be Bayes consistent it has to be part of the set of functions that are proven, in theory, to be universally consistent learners. These are training algorithms that have been shown to approach Bayes optimality, in the limit, as the number of training samples grows to infinity. This includes learning algorithms such as those for generalized linear models (McCullagh, 2019), support vector machines (Steinwart and Christmann, 2008), decision trees (Breiman, 2017), and even certain architectures of neural networks (Faragó and Lugosi, 1993). With these definitions of Bayes optimality, Bayes consistency and universally consistent learners, we can proceed to present the theoretical results for learning prediction functions on incomplete and imputed datasets $\tilde{\mathbf{X}}$ and $\hat{\mathbf{X}}$.

2.3 Missing Data Mechanisms

Before analyzing the theoretical limits of learning prediction functions on datasets with missing values, we must first understand the types of missingness mechanism we may encounter. In his seminal article on *Inference and Missing Data*, Rubin (1976) introduces three types of missingness mechanisms: *Missing Completely at Random* (MCAR), *Missing at Random* (MAR) and *Missing Not at Random* (MNAR). The differences between these three types come from the conditional probability distributions by which the missingness is described.

Assume we would like to quantify the probability of observing a certain missingness pattern $\mathbf{m}$. Then we can describe this probability as $\mathrm{P}(\mathbf{m}|\tilde{\mathbf{x}}, \boldsymbol{\theta})$, where $\tilde{\mathbf{x}}$ is the input vector of partially observed explanatory variables and $\boldsymbol{\theta}$ is a vector of unknown parameters. In the MCAR and MAR settings, this conditional probability can be simplified. The MCAR setting assumes that the probability of missingness is completely independent of the observed and unobserved portions of $\tilde{\mathbf{x}}$. Thus, the probability of observing any missingness pattern $\mathbf{m}$ in the MCAR setting depends solely on the unknown parameters $\boldsymbol{\theta}$, yielding $\mathrm{P}(\mathbf{m}|\boldsymbol{\theta})$. The dependence on the unknown parameters $\boldsymbol{\theta}$ signifies that features are missing randomly, independent of their values or values of the observed features, but there still might be some unknown mechanism that generates certain missingness patterns more often than others. An example to think of here is when a measurement device, *e.g.*, a scale or a sensor, runs out of batteries and fails to record some metric for a certain random sub-sample of the entire dataset.

Under the MAR setting, the probability of any missingness pattern $\mathbf{m}$ reduces to $\mathrm{P}(\mathbf{m}|\tilde{\mathbf{x}}_{obs}, \boldsymbol{\theta})$ signifying the dependence on the observed portion, $\tilde{\mathbf{x}}_{obs}$, of vector $\tilde{\mathbf{x}}$ but not on the unobserved portion, $\tilde{\mathbf{x}}_{mis}$. This is an especially useful assumption for inference tasks, since it allows us to model the values of the missing features $\tilde{\mathbf{x}}_{mis}$ as some function of the observed features $\tilde{\mathbf{x}}_{obs}$ and the unknown parameters $\boldsymbol{\theta}$. An example would be a setting in which we have a camera installed in a forest, and our aim is to take pictures of the animals present in the environment. In addition to the pictures, we also measure humidity and temperature of the air. If a high humidity causes the camera lens to fog up and this in turn causes part of the picture to be considered as missing, the missingness comes from the MAR setting. This is because a fully observed variable caused another variable to be missing regardless of the value of the latter variable.

Finally, under the MNAR setting, the probability of any missingness pattern $\mathbf{m}$ cannot be reduced beyond $\mathrm{P}(\mathbf{m}|\tilde{\mathbf{x}}, \boldsymbol{\theta})$. Importantly, the probability now also depends on the actually missing values of the variables $\tilde{\mathbf{x}}_{mis}$, which makes it impossible to carry out accurate inference. An example would be if a sensor measures heights of certain objects but because it is placed at some height above the ground, the objects lower than this height do not get measured. Then, the missing data essentially undergoes self-censoring, where all the objects under the height at which the sensor is placed do not get measured.

Clearly, the type of missingness mechanism determines whether inference is at all possible. To be able to attempt to use the observed features for inferring the values of the missing ones, the missingness must be generated by the MAR mechanism. Otherwise, the observed values do not contain the information with which we could carry out the task of inference (Rubin, 1976). On the other hand, prediction is possible under any missingness mechanism, including a non-random type of missingness, as in theory we can use a predictor, such as a decision tree, that can natively handle missing values. Alternatively, we could also learn a separate prediction function f for every possible missingness pattern $\mathbf{m}$. However, since missingness pattern is a d-long binary vector, there are up to 2^d distinct missingness patterns in any dataset with d number of variables. Given that the number of potential prediction functions to be learned grows exponentially with the number of variables, the problem of learning them all becomes practically intractable. Moreover, only a small subset of all conceivable missingness patterns is typically observed. Attempting to synthetically generate unobserved patterns to account for potentially new patterns emerging during test time does not accurately reflect real-world conditions, which may lead to further discrepancies between model predictions and actual outcomes. This approach also becomes highly data and memory intensive further decreasing its practicability. Thus in practice, if observations with missing data are not dropped, the missing features are imputed and only a single prediction function f is learned on the imputed dataset. The following section discusses the difficulties in learning this prediction function and its dependence on the missingness mechanism and the type of imputation we use.

2.4 Optimal Learning with Missing Data

Compared with inference, the task of learning optimal predictors, regressors or classifiers, poses a distinct challenge. In the setting of inference one has access to the entire dataset $\tilde{\mathbf{X}}$ and does not have a target vector $\mathbf{y}$. The goal is to accurately infer the missing values in $\tilde{\mathbf{X}}$ producing an imputed dataset $\hat{\mathbf{X}}$ that truthfully reconstructs the distributional properties of the unobserved dataset $\mathbf{X}$. A rich body of literature starting with Rubin (1976) has been produced for this problem. The results from this body of research are, however, not directly applicable to the setting of supervised learning. In the supervised learning setting, the dataset $\tilde{\mathbf{X}}$ is divided into two parts, training set $\tilde{\mathbf{X}}_{train}$ and test set $\tilde{\mathbf{X}}_{test}$, and a target variable vector $\mathbf{y}$, also divided into two sets, $\mathbf{y}_{train}$ and $\mathbf{y}_{test}$, is to be predicted from the imputed datasets. This immediately poses a number of questions different to those posed within the problem of inference. Does one impute both training and test set with the same procedure or should the imputation procedures differ? Should both sets be imputed before training, or should the test set be imputed with some knowledge of the training set acquired during the training process? And, how do these results differ for different types of learners and missingness mechanisms?

2.4.1 Theoretical Results

To begin to answer these questions, we turn to the results first put forth by Josse et al. (2019) who studied the Bayes consistency (2.2) of supervised learning with missing data. They establish the first theoretical results for the consistency of universal learners, those algorithms that can theoretically achieve Bayes optimality (2.2), in the impute-then-regress procedure: first impute the missing values, then train the regressor/classifier. In their paper, in Theorem 3, Josse et al. (2019) show that under the MAR assumption a constant imputation for both training and test set leads to a consistent prediction function f^*_{train} as the number of training samples grows to infinity (the asymptotic regime). This result holds for any per feature imputation value $\alpha_j \in \mathbb{R}, j = 1, \ldots, d$ as long as it is the same for both training and test set. In fact, Josse et al. (2019) show that if α_j is chosen such that it lies outside of the support of the variable j that is being imputed, then the resulting consistency guarantees are even stronger. This is a fairly surprising result as it is in direct contradiction to what accurate imputation aims to achieve in the inference task. Indeed, the result suggests that it is better to impute such that the learner can easily distinguish between a real observation and an imputed one, rather than trying to impute accurately. This is indeed what decision tree-based algorithms do natively. Thus, these result suggests that other algorithms, that do not handle missing data natively, can theoretically do the same, if the missing values remain distinctly recognizable after imputation.

Bertsimas et al. (2021) take this research further, extending the insights from Josse et al. (2019) to encompass all mechanisms of missing data, including MCAR, MAR and MNAR. They also extend the set of imputation functions to include all deterministic imputation strategies such as arbitrary constant imputation, mean imputation, or even more complex approaches such as imputation by a linear combination of the observed features. They demonstrate that, for these deterministic imputation methods, a universally consistent learning algorithm is capable of achieving Bayes consistency. This stems from the fact that such imputation induces a discontinuity in the posterior distribution of the missing variable after imputation. This discontinuity allows the universal learning algorithm to distinguish between imputed and observed variables, facilitating more accurate predictions.

Furthermore, these results suggest that for categorical variables, which are discrete by nature, the mode imputation does not lead to universal consistency. This is because the imputed data points, replaced by the most common category, do not create the same discontinuity in the posterior distribution, making it challenging for the algorithm to distinguish between imputed and observed variables. To address this, Bertsimas et al. (2021) suggest the introduction of a new category specifically for missing instances of categorical variables. This approach effectively marks missing data within the dataset, ensuring that these data points are treated distinctly during the learning process, thereby enhancing the model's ability to learn from both observed and imputed data. Again, the theoretical implications here are clear; imputations that force the learning algorithm to mimic what tree-based methods do natively are preferred over inferentially accurate imputations.

Building upon the foundations (Josse et al., 2019; Bertsimas et al., 2021), Le Morvan et al. (2021) employ a markedly different methodology to derive even more general results. In a series of two articles, Le Morvan et al. (2020, 2021) adopt a topological perspective, imagining a space of all conceivable imputation functions. These functions take as input an arbitrarily long vector of observed values and output the imputed values for the rest of the missing features. Within this topological framework, they establish that for every known mechanism of missingness (MCAR, MAR, NMAR), a universally consistent learner can attain Bayes optimal predictions for *almost all* imputation functions. The term *almost all* is employed in a topological sense relative to the space of functions, indicating that the statement applies to the vast majority of functions within this space, excluding only a negligible subset that does not affect the generality of the conclusion.

Their proof hinges on two key insights. First, they observe that the data points with missing values reside within a lower-dimensional space compared to the fully observed feature vector. This disparity is a critical lever that a universally consistent learner can exploit. However, the leverage is contingent upon the imputation function being designed in such a manner that distinguishes imputed data points from those that were fully observed. This distinction is needed for the learner to effectively utilize the inherent structural differences between complete and incomplete data points, pointing again to the ways in which tree-based models handle missing data natively.

Despite the broad applicability of their findings, Le Morvan et al. (2021) caution against oversimplification. They explain that the performance of the prediction function is dependent on the chosen imputation function. For instance, opting for a straightforward imputation technique like mean imputation might necessitate a prediction function that is discontinuous and lacks smoothness, thereby complicating the learning process with conventional machine learning approaches like gradient descent. Thus, they propose a new neural network architecture called the *NeuMiss* block that imputes missing values. The NeuMiss block can be chained with a regular multi-layered perceptron (MLP) resulting in a neural network that simultaneously optimizes its parameters for missing value imputation and target prediction. This way, the imputation and the prediction functions should adapt to each other, resulting in a better behaved pair of functions that do not experience discontinuities.

2.4.2 Practical Results

The consistency results from the previous section only hold under the assumptions of using universally consistent learning algorithms and a training set with infinitely many data points. Although the first assumption is realizable in practice, by using any algorithm mentioned at the end of Section 2.2, the second certainly is not - we can never have a training dataset with infinitely many data points. Therefore, it is paramount to perform computational experiments that translate the theoretical findings into practice.

In their paper, Le Morvan et al. (2021) perform a computational study on a synthetic dataset of 100,000 training points where they show that the NeuMiss + MLP architecture outperforms all other impute-then-regress procedures on a regression task. Similarly, Bertsimas et al. (2021) confirm their theoretical findings by performing a computational study on 71 curated datasets with synthetically generated missingness patterns (MCAR or MNAR), where they show that the missing category imputation for discrete variables significantly outperforms mode imputation. However, they note that the results are only significant in experiments where the target variable is also synthetically generated - according to a preconceived rule. When the target variable came from a real-world process, the mode vs. missing category imputation did not differ significantly in the downstream regression/classification performance. This naturally raises a question about the applicability of their theoretical results in practice and warrants further empirical experimentation.

Another study by Woźnica and Biecek (2020) evaluates seven different imputation strategies in the impute-then-regress setting on the downstream binary classification task. They use datasets with real missingness and real target signal. Although they show that mean imputation most often outperforms other types of imputations, surprisingly imputing with a random number, picked from the support of the variable that is being imputed, exhibited nearly identical performance. This goes against the theoretical results of Bertsimas et al. (2021) and Le Morvan et al. (2021) as random imputation is not easily de-imputable, hence we would expect lower predictive performance on the downstream task.

Finally, Van Ness and Udell (2023) perform a comparative study between mean imputation in the impute-then-regress setting and three deep imputation models: NeuMiss (Le Morvan et al., 2021), supMIWAE (Ipsen et al., 2022) and GRAPE (You et al., 2020). They use real datasets without missing values and create missingness either according to the MCAR or MNAR assumptions - an approach that we also adapt in our empirical study in Section 4. Across six datasets, they show that the performance on the downstream task does not significantly differ between impute-then-regress procedures with mean imputation and deep imputation models. This result contrasts the synthetic data study performed by Le Morvan et al. (2021), since it suggests that with real datasets, albeit with synthetic missingness, the potential discontinuities in the prediction function after mean imputation do not pose an actual learning problem.

3 Deep Imputation Models

The alternative to impute-then-regress procedures that result in training a regular supervised learning algorithm on the imputed dataset $\hat{\mathbf{X}}$, is developing specialized algorithms that can handle `NaN` entries natively, and thus, be trained directly on $\tilde{\mathbf{X}}$. This section introduces two such algorithms, namely neural networks, that take as input the partially observed data matrix $\tilde{\mathbf{X}}$, within their architecture impute the missing data to produce imputed matrix $\hat{\mathbf{X}}$, and finally use this imputed matrix to generate predictions $\hat{\mathbf{y}}$. Both of the introduced algorithms rely on the basic multi-layered perceptron (MLP) to produce their predictions.

An MLP is a densely connected neural network with some kind of non-linear activation function following every trainable layer (except for the output layer in case of regression). A very small example can be seen in Figure 1, where the prediction block is a small MLP with three trainable layers. The non-linearities, *e.g.*, `ReLu` functions, would follow each of the trainable layers but are not depicted in this figure for simplicity. This MLP is also said to be three layers deep and its trainable layers have width of six, four and one node. It is said to be densely connected because there is a weight parameter, depicted by a line, that can be learned between every pair of nodes in adjacent layers. Finally, a network like this simple MLP can use a procedure of *dropout* for purposes of regularization. If dropout is used, some fraction of nodes during each training loop are dropped and their weights are thus not updated. We adapt this idea of dropout for handling missing values which explains why we call our proposed neural architecture a *Dropout-like* network.

3.1 *Dropout-like* Network

Since deep imputation models have theoretical grounding rooted in adaptation of the imputation function to the prediction function and vice-versa (Le Morvan et al., 2021), we introduce a deep imputation model of our own. Our *Dropout-like* neural network architecture handles missing data natively, imputing the `NaNs` in its first block and predicting the targets in its second block. Figure 1 depicts a schematic view of our model. The first part, *imputation block*, begins by applying conditional dropout to those entries of the input vector $\tilde{\mathbf{x}}$ that are missing - depicted in yellow. The rest of the features, depicted in green, are densely connected to an arbitrarily long and arbitrarily wide MLP section. In Figure 1 this section is one layer long and six nodes wide - its nodes are depicted in grey. The output layer of the imputation block has the same size as the input layer with the originally observed entries of $\tilde{\mathbf{x}}$ being replaced by their true values and the missing nodes, depicted in blue, getting imputed by the output of the MLP section. This produces the imputed vector $\hat{\mathbf{x}}$ which is then directly connected to the second part of our architecture - *the prediction block*. The prediction block is again an arbitrarily long and wide MLP that terminates in the output node.

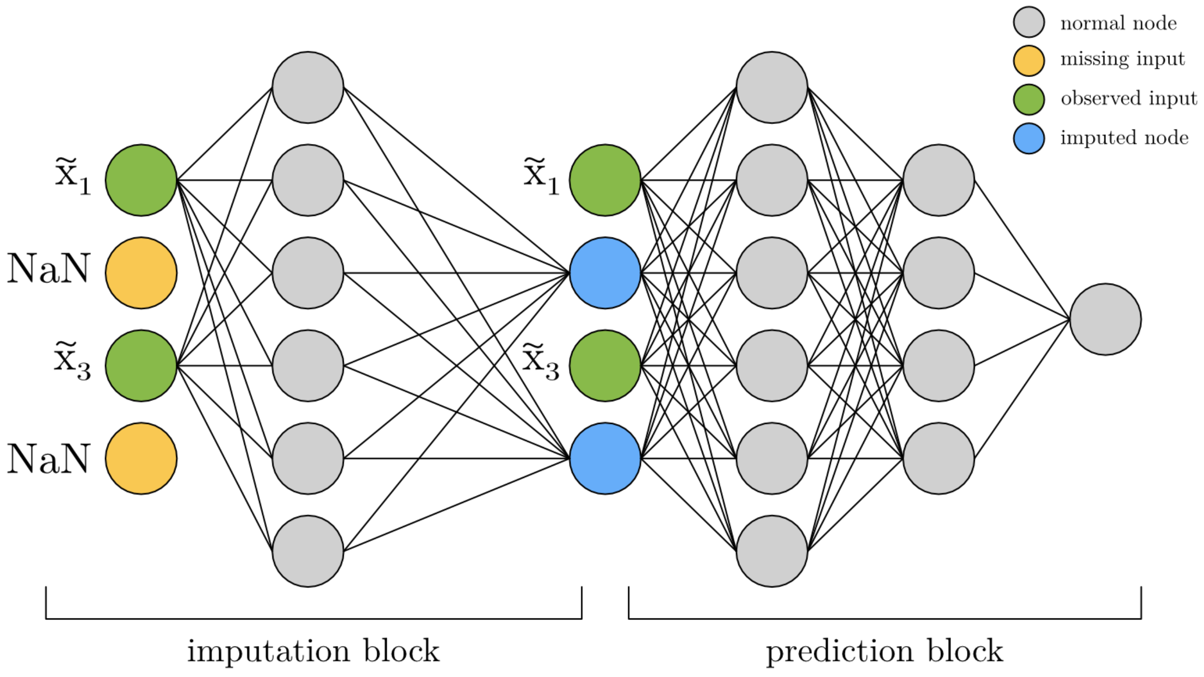

Figure 1: Schematic view of our Dropout-like neural network architecture that natively handles missing data. For simplicity, the bias terms of each layer are omitted.

The conditional masking/dropout of the observed/missing values results in a very simple neural network design that uses only standard components of any basic modern MLP but that can now natively handle missing data. The simplicity of this design also allows for easy extensions of the imputation block to more complex layers such as convolutional layer or transformers. Hence, the idea of conditional dropout is quite universal, lending itself to handle missing data in variety of settings with varying types of inputs (*i.e.*, tabular data, time series or even images).

3.2 NeuMiss Network

In addition to our Dropout-like model, we also describe the NeuMiss MLP introduced by Le Morvan et al. (2020) and further updated in (Le Morvan et al., 2021), since these are the two models that will represent deep imputation methods in our computational study. We choose the NeuMiss network for our computational study because it has the strongest theoretical grounding from the currently proposed deep imputation models in the literature. The NeuMiss MLP, as depicted in Figure 2, also consists of two blocks: an imputation block, called the Neumiss block, and a basic MLP. The Neumiss block is inspired by the approach of iterative algorithm unrolling where an iterative exact algorithm is approximated by a neural network with a specific architecture (Gregor and LeCun, 2010). When this network is trained, its parameters approximate the output of a truncated version of the original algorithm. The Neumiss block implements this idea for the Neumann series, an iterative algorithm used to approximate certain kinds of matrices, *e.g.* matrix inverses. In the context of missing data imputation, the Neumann series is used to approximate a specific linear transformation of the covariance matrix of the input data which is then used to compute the imputed values. Le Morvan et al. (2020) provide a more in depth discussion behind the choice of this specific approximation and the resulting neural network architecture.

However, for a mechanistic understanding of the proposed architecture, Figure 2 recreates its key components. As can be seen, the imputation block has no regular non-linearities such as the `ReLu` function. Instead, the non-linearities are achieved by element-wise multiplication of the input vector with the the missingness mask. The missingness mask, however, uses the opposite notation to the missingness mask introduced in Section 2.1; the observed features are denoted with 1, while the missing features are denoted with 0. The multiplication `NaN` $\times$ 0 results in 0, effectively replacing the `NaN` entries with zeros. This restricts the network to only using the observed values and the trained weights corresponding to these values for generating the imputations. Furthermore, there are only two sets of learnable parameters, a vector $\mathbf{u}$ of length d and a matrix $\mathbf{W}$ of shape $d \times d$, where d is the length of the input vector. Both of these are learned in the backward loop of gradient descent during training. In the forward loop, vector $\mathbf{u}$ is first masked and then subtracted from the already masked input vector. The result of this operation is used as a skip connection that is repetitively added to each iteration of the unrolled Neumann series. The unrolled Neumann series is contained within the Neumiss block, where the multiplication of the matrix **W** with the current version of the input vector is succeeded with the masking operation. This sequence of operations is repeated N times to produce an $N-$approximation of the Neumann series. The result is then fed through a basic feed forward network, resulting in a fully connected network that natively handles missing values.

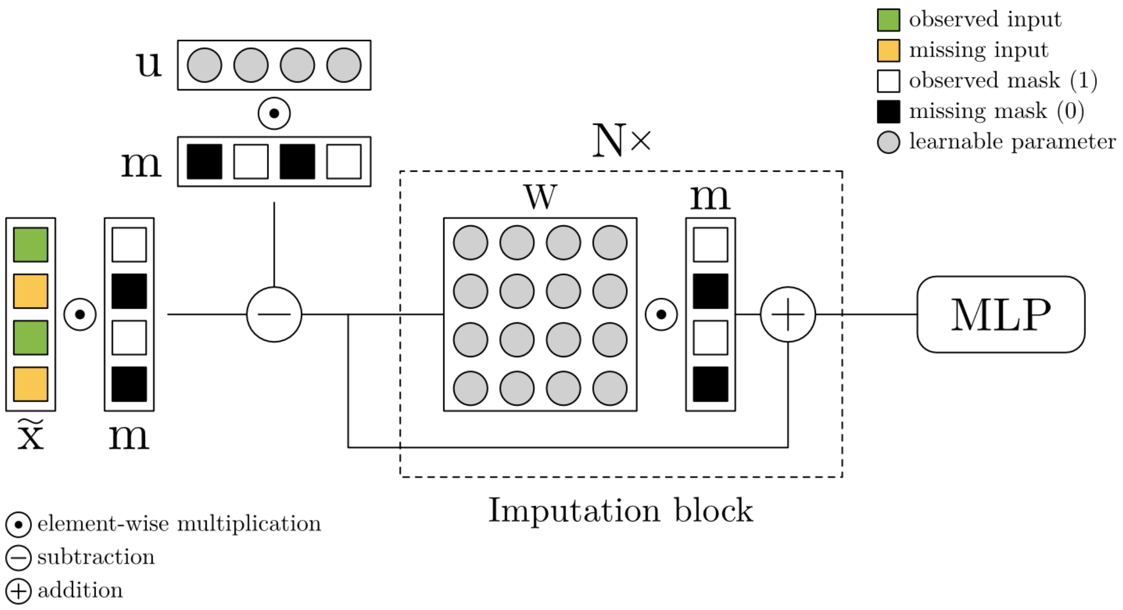

Figure 2: Schematic view of the NeuMiss neural network architecture that natively handles missing data developed by Le Morvan et al. (2020). The element-wise multiplication operates on vectors of the same length and multiplies their entries element-wise. Straight line connections depict regular dot product multiplication. The operations of the imputation block are described in finer detail in Section 3.2.

4 Computational Study

To test how do the recent theoretical results in supervised learning with missing data discussed in Section 2 translate into practice, we carry out an extensive computational study that compares two deep impute models, NeuMiss (Le Morvan et al., 2021) and Dropout-like model, against impute-then-regress pipelines on 10 datasets. In this section we report our experimental setup followed by our findings.

4.1 Datasets

Our experimental setup comprises of 10 different datasets with both synthetic and real missingness mechanisms. Table 2 provides descriptive statistics of five datasets without missing values where we create missingness synthetically either according to MCAR or MNAR assumptions. Under both assumptions we mask as missing at most 10% of the dataset, resulting in various fractions of instances with at least one missing value (see Table 3, column 6). For the MCAR assumption, we randomly pick one third of the features and mask out one third of their entries. To create the MNAR missingness, we pick one fifth of the features and censor both the lower and the upper quartile of the data distribution. The fractions of one third and one fifth are chosen to ensure that at most 10% of the entire dataset is marked as missing after each procedure is applied. Notably, both of these procedures result in at most 7.8% of all possible missingness patterns actually occurring (Table 3, column 8). All datasets with synthetic missingness come from the OpenML repository. The context from which the datasets come are varied. The `christine` and `philippine` datasets are pre-processed real-life datasets from an Auto-ML competition where participants were asked to develop automatic prediction pipelines without the need for human intervention. The `airfoil` dataset comes from NASA and the data represents a series of aerodynamic and acoustic tests. Finally, the `winequality` dataset contains numerical features of wines such as acidity or sugar content, the target is the average wine tasters' rating.

For datasets with real missingness, we report their descriptive statistics in Table 3. Here, the datasets `wiki` and `support` come from the UCI repository, while the datasets `equity, fico` and `cirrhosis` come from the OpenML repository. The datasets represent a wide variety of data types ranging from textual features in the `wiki` case, through financial datasets in the `equity` and `fico` cases, to medical data in the `support` and `cirrhosis` instances. The fraction of missing values in these datasets ranges from 1.5% to 9.3%, while the fraction of samples that have at least one missing feature ranges between 31.3% and 96.6%. Notably, in all datasets only a negligible fraction of all possible missingness patterns actually occurs. Moreover, the inclusion of datasets with real missingness patterns is a novel addition to a computational study of this kind, since all previous work only looked at synthetically induced missingness (Bertsimas et al., 2021; Van Ness and Udell, 2023). All datasets, including the datasets with synthetically masked out missing values, can be downloaded from our GitHub repository [1].

[1]`https://github.com/TomasMiskov/missing_imputation_thesis_code`

Dataset	Target	n	d	d/n	Fraction missing	Fraction of instances with missing	Missing patterns	Fraction of missing patterns
christine (MCAR)	Class.	5418	1637	0.302	0.109	0.330	1787	0*
philippine (MCAR)	Class.	5832	309	0.053	0.108	0.330	1924	0*
airfoil (MCAR)	Reg.	1503	6	0.004	0.055	0.329	5	0.078
winequality (MCAR)	Reg.	6497	12	0.002	0.082	0.330	165	0.040
phoneme (MCAR)	Class.	5404	6	0.001	0.055	0.330	5	0.078
christine (MNAR)	Class.	5418	1637	0.302	0.093	1.000	5418	0*
philippine (MNAR)	Class.	5832	309	0.053	0.098	1.000	5832	0*
airfoil (MNAR)	Reg.	1503	6	0.004	0.083	0.497	1	0.016
winequality (MNAR)	Reg.	6497	12	0.002	0.080	0.725	3	0.001
phoneme (MNAR)	Class.	5404	6	0.001	0.083	0.500	1	0.016

Table 2: Descriptive statistics of datasets with synthetically created missing values. Here, n stands for the number of samples and d for the number of features. The fraction of missing patterns is calculated as the number of observed missing patterns divided by 2^d. The values 0* occur due to rounding.

Dataset	Target	n	d	d/n	Fraction missing	Fraction of instances with missing	Missing patterns	Fraction of missing patterns
support	Class.	1000	50	0.050	0.093	0.966	310	0*
equity	Class.	5960	19	0.003	0.042	0.410	74	0*
fico	Class.	9871	38	0.004	0.034	0.747	74	0*
cirrhosis	Class.	1945	22	0.011	0.030	0.570	9	0*
wiki	Class.	904	67	0.074	0.015	0.313	157	0*

Table 3: Descriptive statistics of datasets with real missing values. Here, n stands for the number of samples and d for the number of features. The fraction of missing patterns is calculated as the number of observed missing patterns divided by 2^d. The values 0* occur due to rounding.

4.2 Study Design

In our study, we evaluate the performance of five different algorithms to handle missing data: two impute-then-regress pipelines, one tree-based model and two deep impute models. For the impute-then-regress pipelines, we first fill in the missing values using mean imputation and then train two types of models: a basic MLP and a random forest model. We choose the basic MLP to include an impute-then-regress model that is highly comparable to the two deep impute networks introduced in Section 3. A random forest model is also included, namely because of its ability to fit highly complex prediction functions which makes it a strongly performing alternative to neural networks. Moreover, because random forest is a tree-based model, it can also handle `NaNs` natively. We thus train two distinct random forests, one with the imputed dataset $\hat{\mathbf{X}}$ as part of the impute-then-regress procedure, and one with the original, partially observed dataset $\tilde{\mathbf{X}}$ as a standalone comparison to the deep imputation networks.

For the basic MLP, we choose to have two hidden layers of widths $2^{[\log_2(d)+1]}$ and $2^{[\log_2(d)]}$, where [ ] denotes the nearest integer function and d is the dimension of the dataset. The width of the layers is thus an exponent of two and this choice is made because we will be training the network on a GPU. This practice helps to align the architecture with the GPU's parallel processing capabilities, increasing the computational efficiency during training. Moreover, the input layer is of size d and output of size one. Each layer, except the final one, is followed by a `ReLu` activation function and the parameters of the network are initialized according to Kaiming He initialization to improve learning speed and stability (He et al., 2015). The loss function is mean squared error (MSE) for regression and binary cross entropy (BCE) with sigmoid activation for binary classification - the two supervised prediction tasks we consider in our study. We use the Adam optimizer with fixed learning rate of 10^{-4}, and we train the network for 5,000 epochs, stopping early if no improvement in the validation loss occurs for 100 consecutive epochs. The training set is always split such that there are at most 10 batches of training data per epoch, resulting in comparable number of parameter updates per epoch per dataset. Both the network architecture and the training loop are implemented in PyTorch (Paszke et al., 2019) and are available on our GitHub page. The random forest regressor/classifier is trained with parameters as suggested by Probst et al. (2019) in Table 1 and it is implemented using the Scikit Learn library (Pedregosa et al., 2011). Since Scikit Learn does not natively support handling of missing entries for random forest models, we use it only for fitting random forests on the mean imputed datasets $\hat{\mathbf{X}}$. For the alternatively trained random forest on the un-imputed datasets $\tilde{\mathbf{X}}$, we use the implementation provided in the LightGBM package (Ke et al., 2017, Table 1).

For deep impute models, we train a NeuMiss network and a Dropout-like network without performing any imputations. To allow for as fair of a comparison as possible, we keep the prediction blocks of both types of networks the same as the basic MLP. Both NeuMiss and a Dropout-like networks thus only differ in their imputation block. For the Dropout-like network, we add one hidden layer of size $2^{[\log_2(d)+1]}$ to its imputation block, followed by a layer of size d where the imputations are integrated with the observed values, followed by a basic MLP. For the NeuMiss network, we choose the depth of the imputation block to be 30, which should provide adequate trade off between performance and computational training cost and is in line with depths used in (Le Morvan et al., 2020). The NeuMiss imputation block is then again followed by the basic MLP. Both deep imputation networks are implemented in PyTorch (Paszke et al., 2019) and trained with the same training loop as described in the previous paragraph.

For every dataset in our study, we conduct five separate experiments using different random seeds. In each experiment, the dataset is divided into training, validation, and test sets with ratios of 64, 16 and 20 percent respectively. We then standardize each feature based on the descriptive statistics, mean and standard deviation, of the training set. For the impute-then-regress pipelines, we impute all missing values across all subsets of data with zeros. This approach is equivalent to training set mean imputation before normalization and scaling. Once the datasets are prepared, we proceed to train the respective five models and record their training times alongside the relevant regression/classification metrics on the holdout test set. For regression, we report the R-squared score that captures the fraction of variance in the target variable that can be explained by our model. For classification, we report the F1-score that provides a balanced metric between precision and recall. To find out if the models yield significantly different scores from each other, we perform the Friedman non-parametric test which is an alternative to repeated measures ANOVA. We choose this test because it is based on model rankings and does not assume normality which we cannot guarantee with only five repeated measures of the model's performance on each dataset. If the Friedman test produces a significant test statistic, we carry out a post-hoc analysis with a Conover-Friedman test that yields p values for every pair of tested models. The p values are adjusted for multiple comparisons with the Bonferroni adjustment. All experiments are carried out on a laptop computer with 32GB of RAM, 2.60GHz Intel(R) Core(TM) i9-13900H CPU and Nvidia GeForce RTX™ 4060 GPU. The training of neural networks in Pytorch is performed with `cuda` on the GPU.

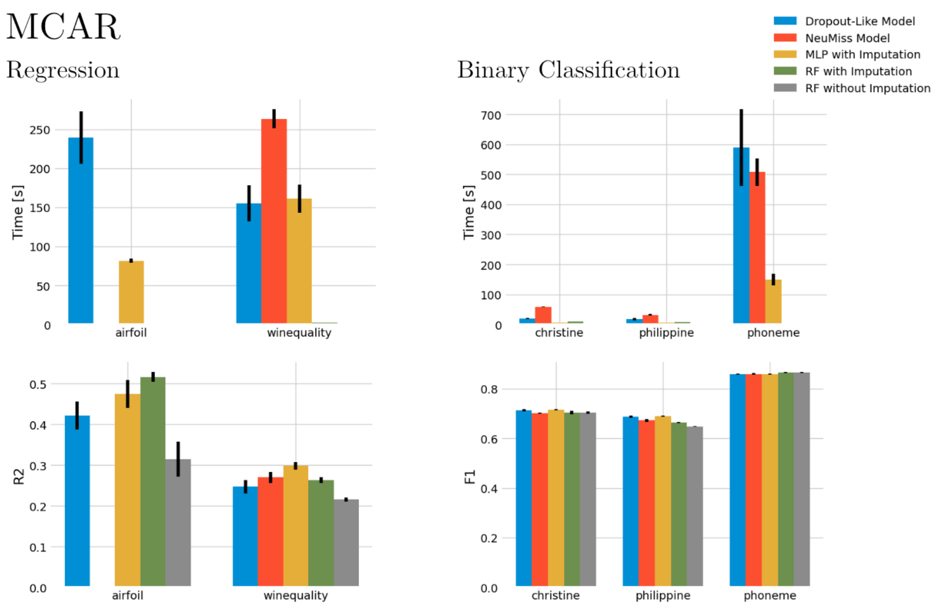

Figure 3: Training times, R-squared scores and F1 scores for the MCAR datasets. The bars of the training times for most of the random forest models are not visible due to an order of magnitude faster fitting times than neural networks. The bars for the NeuMiss model are missing for the `airfoil` dataset as the model did not converge within 5000 epochs of training. Error bars are based on standard errors of five trials with different random seeds.

4.3 Results

Figures 3, 4 and 5 show the results of our computational study for datasets with MCAR, MNAR and real missingness respectively. All figures report the training times and either R-squared score, if the task was regression, or F1 score, if the task was classification. For both metrics, the higher the bars, the better the performance. The best possible performance for both metrics is a score of one. Furthermore, in Tables 4, 5, and 6, we report the p values of the Conover-Friedman post-hoc tests for those datasets within real, MCAR, and MNAR experiments respectively, where the Friedman statistic indicated significant differences between the models.

For the experiments with MCAR missingness, we can see in Figure 3 that the deep impute models did not outperform the impute-then-regress pipelines or the random forest model without any imputations. In the datasets with regression tasks, left side of the figure, either the MLP with imputations or the random forest with imputations produced the highest R-squared scores on average. Table 5 reports the p values of the Conover-Friedman test which indicates that some of these differences in R-squared scores were indeed significant. For the classification tasks, right side of the figure, the results in F1 scores did not produce significant differences for the `christine` and `phoneme` datasets where all models achieved comparable performance. In the `philippine` dataset, the random forest without imputations was found to be significantly under performing, however, no significant results were found between any of the deep impute models and the impute-then-regress pipelines.

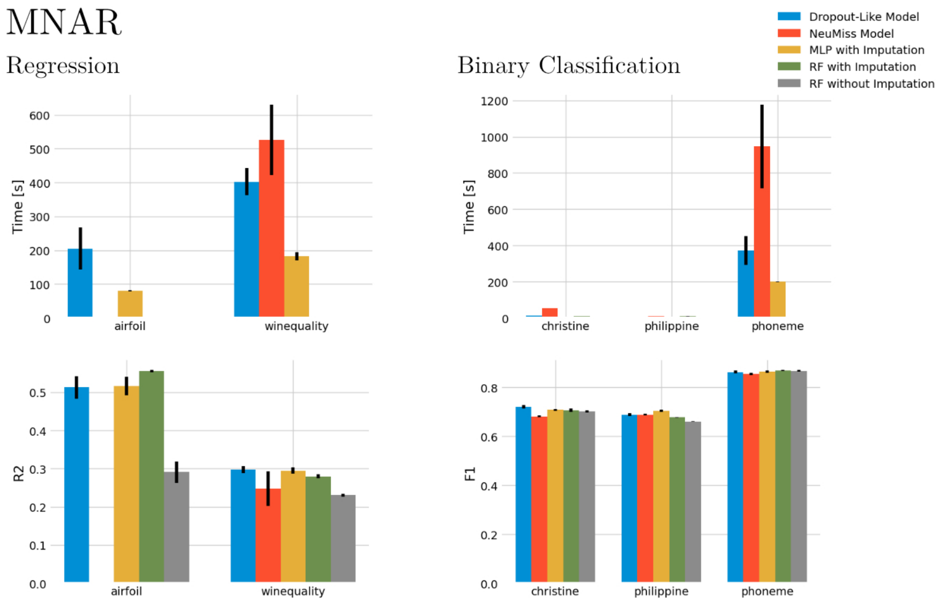

Figure 4: Training times, R-squared scores and F1 scores for the MNAR datasets. The bars of the training times for most of the random forest models are not visible due to an order of magnitude faster fitting times than neural networks. The bars for the NeuMiss model are missing for the `airfoil` dataset as the model did not converge within 5000 epochs of training. Error bars are based on standard errors of five trials with different random seeds.

Although very similar to the MCAR case, the results for the MNAR missingness seen in Figure 4 produced some significant differences between deep impute models and impute-then-regress pipelines as evidenced in Table 6. These differences occurred in the `airfoil` and the `christine` datasets, where one of the impute-then-regress pipelines outperformed one of the deep impute models. The rest of the results are non-significant, except for the generally worse performance of the random forest without imputations.

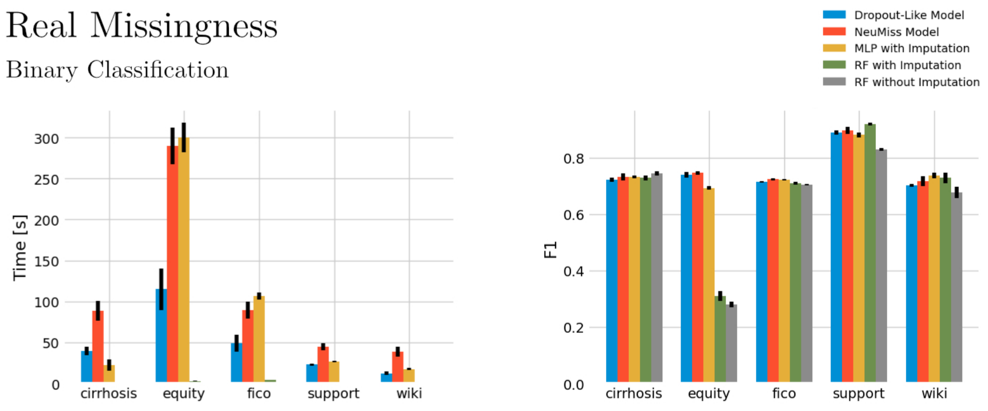

Figure 5: Training times and F1 scores for datasets with real missingness. The bars of the training times for most of the random forest models are not visible due to an order of magnitude faster fitting times than neural networks. Error bars are based on standard errors of five trials with different random seeds

Finally, in the results of the experiments with real missingness, seen in Figure 5 and Table 4, one or both of the deep impute models significantly outperformed one or both of the impute-then-regress pipelines in the `equity` and `fico` datasets. In the remaining three datasets, the resulting performance between these two categories of models was non-significantly different. Moreover, as with MCAR and MNAR missingness, the random forest without imputations significantly under-performed in `equity, fico` and `support` datasets. However, this model also achieved the highest average score on the `cirhosis` dataset, albeit non-significant.

Real Missingness

equity

	🟦	🟥	🟨	🟩	⬛
🟦	1.000				
🟥	1.000	1.000			
🟨	0.141	0.141	1.000		
🟩	**0.000**	**0.000**	**0.021**	1.000	
⬛	**0.000**	**0.000**	**0.008**	1.000	1.000

fico

	🟦	🟥	🟨	🟩	⬛
🟦	1.000				
🟥	0.160	1.000			
🟨	0.160	1.000	1.000		
🟩	0.160	**0.001**	**0.001**	1.000	
⬛	0.160	**0.001**	**0.001**	1.000	1.000

support

	🟦	🟥	🟨	🟩	⬛
🟦	1.000				
🟥	1.000	1.000			
🟨	1.000	0.530	1.000		
🟩	0.132	1.000	**0.031**	1.000	
⬛	**0.031**	**0.002**	0.132	**0.000**	1.000

Table 4: P values of Conover-Friedman post-hoc test for those datasets that had significant Friedman test statistic (p value < 0.05) under experiments with real missingness. The colors represent models as stated in Figure 5. Blue: Dropout-like model, red: NeuMiss model, yellow: MLP with imputation, green: Random forest with imputation, grey: Random forest without imputation. Significant differences between two models are depicted in **bold**.

MCAR

airfoil

	🟦	🟥	🟨	🟩	⬛
🟦	1.000				
🟥					
🟨	0.066		1.000		
🟩	0.167		**0.001**	1.000	
⬛	**0.026**		**0.000**	1.000	1.000

philippine

	🟦	🟥	🟨	🟩	⬛
🟦	1.000				
🟥	0.675	1.000			
🟨	1.000	0.064	1.000		
🟩	0.064	1.000	**0.005**	1.000	
⬛	**0.000**	**0.012**	**0.000**	0.144	1.000

winequality

	🟦	🟥	🟨	🟩	⬛
🟦	1.000				
🟥	1.000	1.000			
🟨	0.107	1.000	1.000		
🟩	1.000	1.000	0.604	1.000	
⬛	0.604	**0.059**	**0.002**	0.107	1.000

Table 5: P values of Conover-Friedman post-hoc test for those datasets that had significant Friedman test statistic (p value < 0.05) under experiments with MCAR missingness. The colors represent models as stated in Figure 3. Blue: Dropout-like model, red: NeuMiss model, yellow: MLP with imputation, green: Random forest with imputation, grey: Random forest without imputation. Significant differences between two models are depicted in **bold**.

MNAR

airfoil

	🟦	🟥	🟨	🟩	⬛
🟦	1.000				
🟥					
🟨	**0.036**		1.000		
🟩	1.000		**0.036**	1.000	
⬛	1.000		**0.005**	1.000	1.000

christine

	🟦	🟥	🟨	🟩	⬛
🟦	1.000				
🟥	**0.000**	1.000			
🟨	1.000	**0.003**	1.000		
🟩	1.000	**0.010**	1.000	1.000	
⬛	**0.042**	0.330	0.330	1.000	1.000

philippine

	🟦	🟥	🟨	🟩	⬛
🟦	1.000				
🟥	1.000	1.000			
🟨	0.144	0.144	1.000		
🟩	0.675	0.675	**0.002**	1.000	
⬛	**0.002**	**0.002**	**0.000**	0.144	1.000

winequality

	🟦	🟥	🟨	🟩	⬛
🟦	1.000				
🟥	1.000	1.000			
🟨	1.000	1.000	1.000		
🟩	1.000	1.000	1.000	1.000	
⬛	**0.050**	0.389	**0.030**	0.389	1.000

Table 6: P values of Conover-Friedman post-hoc test for those datasets that had significant Friedman test statistic (p value < 0.05) under experiments with MNAR missingness. The colors represent models as stated in Figure 4. Blue: Dropout-like model, red: NeuMiss model, yellow: MLP with imputation, green: Random forest with imputation, grey: Random forest without imputation. Significant differences between two models are depicted in **bold**.

5 Discussion

On the one hand, the recent theoretical results prove that for specific types of missingness (Josse et al., 2019; Bertsimas et al., 2021), and for any type of missingness in general (Le Morvan et al., 2021), the constant imputation function allows a Bayes consistent learner (*e.g.*, a decision tree) to achieve Bayes optimality. On the other hand, the function to be learned may be non-smooth and non-continuous (Le Morvan et al., 2021), posing real practical challenges in estimating it from a finite sample of training points. This makes deep impute models attractive, since joint optimization of imputation and prediction should adopt one function to the other and vice versa. *Does this assertion hold in practice?*

As our experiments show, the above reasoning did not hold up to the practical scrutiny. Figures 3, 4 and 5 tell a convincing story about deep impute models being merely as good, if not worse, than the impute-then-regress pipelines. The only exception to this observation is the `equity` dataset (Figure 5), where the two deep impute models clearly outperformed the rest of the alternatives. However, in the remainder of the experiments, the two competing approaches produced comparable results. And even if one or the other showed significant difference based on the Conover-Friedman tests, the magnitude of this difference can hardly be considered meaningful in practice. This is especially the case for the classification tasks under the synthetically created MCAR and MNAR missingness (right side of the Figures 3 and 4). The magnitudes on these tasks hardly differ between the five tested models which is further confirmed by the non-significant Conover-Friedman statistics. Only in one out of six cases was there a significant difference between one of the deep impute models and one of the impute-then-regress pipelines. Moreover, in three out of six cases, the random forest model that was trained on the non-imputed datasets performed significantly worse than at least one of the other models. This subset of our experiments thus suggests that mean imputation is better than no imputation for random forests, but deep impute models do not outperform impute-then-regress pipelines.

For the regression tasks under MCAR and MNAR missingness (left side of the Figures 3 and 4), the magnitudes of the differences between the models are bigger than for the classification task. However, these differences are in favor of impute-then-regress pipelines rather than the deep impute models. In three out of four cases, one of the impute-then-regress models achieved better average score than the deep impute networks. Twice was this result significant and four out of four times the random forest trained on non-imputed datasets achieved significantly lowest score. These results suggest that for synthetic missingness mechanisms, the assertions of deep impute models being superior to impute-then-regress pipelines as proposed by Le Morvan et al. (2021) do not prove true in our empirical study.

When it comes to real missingness, Figure 5 and Table 4, the `cirrhosis` and `fico` datasets show negligible, non-significant differences between the five models. For the remaining three datasets, only once the deep impute models clearly, significantly outperform the impute-then-regress pipelines. For the other two cases, once the random forest model trained on imputed datasets significantly outperforms the NeuMiss model, and once the differences are non-significant but the impute-then-regress pipelines achieve higher F1 scores on average. Hence, even in real missingness, the assertion of deep impute networks yielding better predictive performance do not hold up to our empirical scrutiny.

Overall, our results contrast the results obtained by Le Morvan et al. (2021) where they showed their NeuMiss network outperforming other methods. In our experiments, the NeuMiss network merely matches the performance of the other models, showing that the results from synthetically generated data do not readily translate to practice. Moreover, the Dropout-like network proposed by us also matches the performance of the NeuMiss network, albeit at generally shorter training times. Thus if training time is of importance, the Dropout-like network could be prefered by practitioners over the NeuMiss architecture. However, since the random forest model trained on imputed data achieved in most cases comparable if not better performance, under time constraints this should be the preferred model.

On the topic of training time, we also find large inter-dataset variability in training times for the neural network models. This variability is not directly related to the size of the dataset or the number of variables. Indeed, the training times for the `phoneme` dataset are much longer than for `christine` or `philippine` while the number of training samples is approximately the same, and `phoneme` has only 6 variables compared with 309 and 1637 of `christine` and `philippine` respectively. We hypothesize that the much longer training times come from the difficulty of the actual predictive task itself, since the validation loss has decreased for 5000 epochs for the `christine` dataset, while early stopping has been triggered for both `christine` and `philippine` within the first few hundred epochs. The `equity` dataset also resulted in especially long training times in addition to very poor performance of both random forest models. Since it was not the largest dataset in terms of training instances or variables (see Table 3), we likewise hypothesize that the observed results are due to the difficulty of the prediction task rather than due to the size of the dataset.

Finally, although fairly convincing, the obtained results should be extrapolated with caution, since all of the obtained metrics are based on merely five repeated measures and tabular datasets only. Moreover, as the results from the `equity` dataset suggest, there might indeed exist some datasets in which the deep impute models produce significantly better results. Therefore, making general claims about the lack of practical performance of the deep impute models is not warranted from our computational study. Moreover, the considered datasets are fairly small in size for today's standards. Performing our experiments on datasets with much larger number of instances, where the neural networks would have many more observations to fit their highly flexible prediction functions, could substantially alter the results. Nevertheless, the empirical evidence presented here sheds some light on the translatability of theoretical results in supervised learning with missing data to practice.

6 Conclusion

This thesis has put to practical scrutiny the recent theoretical results in supervised learning with missing data. We find that departing from the synthetically generated data and synthetic missingness mechanisms, the theoretical results do not hold up to our test in real-world applications. Although our study only tested 10 datasets, five models and performed five repeated measure of performance per model, the results clearly contrast the theoretical expectations of deep impute models outperforming other approaches. This points to the fact that the non-smoothness and non-continuity in the Bayes optimal prediction function that is created by using constant mean imputation as suggested by (Le Morvan et al., 2021), may not be as severe as assumed in theory. For this reason, the impute-then-regress pipelines can readily learn this function and thus match the performance of deep impute models that impute in their own way, but this way does not allow the models to find a better performing predictive function. This highlights the important practical implication, namely that for tabular datasets with up to a few thousand data points, the impute-then-regress pipelines with constant mean imputation generally provide competitive predictive performance to the deep imputation networks for a fraction of their training time. We hope that this work highlights the importance of testing the theory in practice and gives a useful guideline for fitting supervised predictive models on datasets with missing values. In the future, the deep impute models should be put to further scrutiny in a more extensive computational study, testing larger amount of datasets and exploring what features of the data result in deep impute models outperforming impute-then-regress pipelines and vice versa. It is entirely possible that the results in our study are dependent on the datasets we tested and in a larger study the fraction of datasets with deep impute models outperforming the rest would be higher. Beyond larger empirical testing of the recent theoretical results presented in this thesis, further theoretical strides can still be made in this field. Answering questions regarding the optimal architecture of the deep impute models or why is there a stark performance difference between random forest models trained on imputed and non-imputed data, would certainly be of interest to both researchers and practitioners.

References

Azur, M. J., Stuart, E. A., Frangakis, C., and Leaf, P. J. (2011). Multiple imputation by chained equations: what is it and how does it work? *International Journal of Methods in Psychiatric Research*, 20(1):40–49.

Bertsimas, D., Delarue, A., and Pauphilet, J. (2021). Prediction with missing data. *Stat*, 1050:7.

Breiman, L. (2017). *Classification and Regression Trees.* Routledge, New York, NY, 1st edition.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of The Royal Statistical Society: Series B (Methodological)*, 39(1):1–22.

Devroye, L., Györfi, L., and Lugosi, G. (1996). *A Probabilistic Theory of Pattern Recognition.* Springer Science & Business Media, New York, NY, 1st edition.

Faragó, A. and Lugosi, G. (1993). Strong universal consistency of neural network classifiers. *IEEE Transactions on Information Theory*, 39(4):1146–1151.

Gregor, K. and LeCun, Y. (2010). Learning fast approximations of sparse coding. In *Proceedings of the 27th International Conference on Machine Learning*, pages 399–406.

Grinsztajn, L., Oyallon, E., and Varoquaux, G. (2022). Why do tree-based models still outperform deep learning on typical tabular data? *Advances in Neural Information Processing Systems*, 35:507–520.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1026–1034.

Ipsen, N. B., Mattei, P.-A., and Frellsen, J. (2022). How to deal with missing data in supervised deep learning? In *10th International Conference on Learning Representations.*

Josse, J., Prost, N., Scornet, E., and Varoquaux, G. (2019). On the consistency of supervised learning with missing values; 2019. *Available from: arXiv*, 1902.06931. Last accessed on 18th April 2024.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., and Liu, T.-Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. *Advances in Neural Information Processing Systems*, 30.

Le Morvan, M., Josse, J., Moreau, T., Scornet, E., and Varoquaux, G. (2020). Neumiss networks: differentiable programming for supervised learning with missing values. *Advances in Neural Information Processing Systems*, 33:5980–5990.

Le Morvan, M., Josse, J., Scornet, E., and Varoquaux, G. (2021). What's a good imputation to predict with missing values? *Advances in Neural Information Processing Systems*, 34:11530–11540.

Lehmann, E. L. and Casella, G. (1998). *Theory of Point Estimation.* Springer Science & Business Media, New York, NY, 2nd edition.

McCullagh, P. (2019). *Generalized Linear Models.* Routledge, New York, NY.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. *Available from: arXiv*, 1912.01703. Last accessed on 18th April 2024.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Probst, P., Wright, M. N., and Boulesteix, A.-L. (2019). Hyperparameters and tuning strategies for random forest. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 9(3):e1301.

Rubin, D. B. (1976). Inference and missing data. *Biometrika*, 63(3):581–592.

Śmieja, M., Struski, Ł., Tabor, J., Zieliński, B., and Spurek, P. (2018). Processing of missing data by neural networks. *Advances in Neural Information Processing Systems*, 31.

Steinwart, I. and Christmann, A. (2008). *Support Vector Machines.* Springer Science & Business Media, New York, NY.

Sun, Y., Li, J., Xu, Y., Zhang, T., and Wang, X. (2023). Deep learning versus conventional methods for missing data imputation: A review and comparative study. *Expert Systems with Applications*, page 120201.

Van Ness, M. and Udell, M. (2023). In defense of zero imputation for tabular deep learning. In *Advances in Neural Information Processing Systems 2023 Second Table Representation Learning Workshop.*

Woźnica, K. and Biecek, P. (2020). Does imputation matter? benchmark for predictive models. *arXiv preprint arXiv:2007.02837.* Last accessed on 18th April 2024.

You, J., Ma, X., Ding, Y., Kochenderfer, M. J., and Leskovec, J. (2020). Handling missing data with graph representation learning. *Advances in Neural Information Processing Systems*, 33:19075–19087.