Master's Thesis
Course of Studies: Computer Science

Assessing and Predicting the Optimal Imputation Method Regarding the Predictive Performance of Machine Learning Models

by

Pascal Dittrich

85521

Supervising Professor:	Prof. Dr. Martin Heckmann
Co-Supervisor:	M. Sc. Stefan Wehrenberg
Submission Date:	26. May 2023

Statutory Declaration

I, **Pascal Dittrich**, hereby declare that I have wrote the information available in this work truthfully and independently by myself.

Furthermore, I assure that I have used no other than the specified sources and aids, that I have marked all quotations that I have used from other sources, and that the work in the same or similar was not yet part of a study or examination.

Location, Date Signature (Student)

Abstract

With the increasing demand for high-quality data for AI applications, the issue of incomplete and corrupted data has become a reoccurring and time-consuming problem for data scientists and practitioners. Ensuring completeness is a central requirement for data that is used to train robust and reliable machine learning models. An increasingly popular way to deal with incomplete datasets is missing data imputation. While there are several methods and approaches to consider, this thesis aims to benchmark six proven imputation methods regarding their impact on the predictive performance of machine learning models. The imputation approaches investigated in this thesis are mean/mode, K-NN, random forest, discriminative deep learning, VAE, and GAIN. This thesis also facilitates a large number of datasets and a variety of different, realistic settings to corrupt these datasets with missing values, which in turn are imputed by the aforementioned imputation methods. The imputed data is used to train and test machine learning models, covering binary classification, multiclass classification, and regression.

The results regarding the predictive performance of the imputation experiments are compared to the performance of models trained and tested on uncorrupted data, as well as to the predictive performance of models trained and tested on imputed subsets of the datasets. Additionally, a model is trained on the dataset properties of the conducted imputation experiments and the imputation methods, that are deemed as best suited for each of the different experiment scenarios. This model aims to suggest the best suited imputation method for a given corrupted dataset.

The key findings of this thesis indicate, that random forest represents on average the best imputation approach. Several approaches to automate or simplify the decision-making in regard to the best suited imputation method yield on average no significant improvement to the performance of this imputation approach.

Contents

List of Figures

List of Tables

Listings

List of Abbreviations

MNAR Missing Not at Random

MCAR Missing Completely at Random

MAR Missing at Random

GAIN Generative Adversarial Imputation Nets

VAE Variational Autoencoder

K-NN K-Nearest Neighbors Algorithm

MICE MICE

SVM Support Vector Machine

SVR Support Vector Regression

CSV Comma Separated Values

NAN Not a Number

SGD Stochastic Gradient Descent

PCA Principal Component Analysis

GPU Graphics Processing Unit

RMSE Root Mean Squared Error

NASA National Aeronautics and Space Administration

API Application Programming Interface

ID Identity

1. Introduction

In 2006 the mathematician Clive Humby proclaimed that data is the new oil (Viernes, 2022), and since then the phrase has been reused multiple times over by a variety of people. While there are certain distinctions between the importance of oil and data, there are few people in the age of digitalization who would dispute the ever-growing importance of the latter.

Technological advances in areas such as “Big Data” or “Artificial Intelligence” require huge amounts of data. But not only the quantity of data is of relevance, the quality of the available data is just as important. Insufficient data quality can lead to several problems, most recent prominent example of this is the popular debate around the question, of whether AI can be considered racist (Wolf, 2023). Incomplete and unbalanced data can lead to this kind of biased outcomes for artificial intelligence applications.

In order to avoid these kinds of problems, data scientists spend the majority of their time preprocessing the data they are given (Munson, 2012). Their expertise is vital for the performance of the applications and analysis based on their preprocessed data. This leads eventually to a situation where the data preprocessing becomes a bottleneck for the development and usage of AI-based technology. This thesis aims to support data scientists, researchers, and practitioners by researching how to deal with missing data in incomplete datasets, with a focus on data imputation.

1.1. Problem Definition and Research Objectives

A reoccurring problem, that data scientists and practitioners frequently face, is missing data or simply incomplete datasets. While there is a variety of reasons for which data is missing, the outcome for the usage of this data is usually similar. With incomplete datasets, it is difficult to develop AI-based technologies, train machine learning models and conduct high-quality data analysis. In order to deal with this issue, data scientists have two options at hand. Either they are deleting data that is incomplete or they utilize data imputation to fill in the gaps. Since the deletion of data always leads to a further loss of information, the latter approach is usually preferred.

While it sounds rather easy to simply suggest data imputation, there is a lot to

consider when taking this approach. Based on factors like the size and shape of the dataset at hand, the amount of missing data, a potential pattern behind that missingness, or whether categorical or numerical data is missing, an informed and educated decision about the exact method and advance for the imputation needs to be made. Additionally, it is important to consider, how this imputation will also affect the further usage of the data, for instance, the training of a downstream machine learning model and in turn the predictive performance of said mode. To make such a decision, a lot of experience and expertise is required.

That is the part, where this thesis attempts to support researchers and practitioners by evaluating the impact of data imputation on training and test data for the predictive performance of a downstream machine learning model, which is trained and tested on those imputed datasets. Will the performance of the downstream model increase or decrease through the imputation of the missing data? Does it make a difference which data imputation method is used and if it does, is there a way to figure out the best method for the dataset and task at hand?

In order to answer these questions, six different imputation approaches will be tested for a variety of different scenarios and datasets for three different downstream machine learning tasks. This thesis builds on the work of Jäger et al. (2021) and uses most of the experiment settings facilitated in their research. The target is to identify differences between these six imputation methods for the described scenarios and, if available, identify a best imputation method. Assuming different imputation methods are better suited for certain scenarios, further research is investigating ways to determine the best suited imputation approach for a given incomplete dataset at hand. These potential ways to determine the best suited approach aim to make the lives of scientists and practitioners easier in the future.

1.2. Apporach for this Thesis

To answer the described questions and reach the target for this thesis, the following approach is taken. After outlining the targets for this thesis, a literature research is conducted. The work of Schelter et al. (2021) and Jäger et al. (2021) is reviewed in more detail, especially the experiments and results from the latter. After a short explanation of the methodology applied for this thesis, the author continues to explain the different experiments conducted in this thesis. The following chapter describes the results of those experiments in detail. The penultimate chapter concludes and discusses the highlights and key findings of the thesis. In the last chapter, the conducted activities and the gained knowledge from this thesis are shortly summarized and an outlook for future research is provided.

2. Theoretical Background

2.1. Data Quality

Data Quality is an essential topic for anyone involved in Data Science or Artificial Intelligence. Generally, it is preferable to have as much data as possible and ideally the available data should reach a certain level of quality. This quality level can be measured in several different ways, which vary in importance depending on the required task or purpose. The Data Management Association in the UK proposes therefore six central characteristics to determine the quality of data, as described by the British Government Data Quality Hub (2021):

- Accuracy
- Completeness
- Consistency
- Uniqueness
- Timeliness
- Validity (Hub, 2021)

According to Gawande (2021), accuracy is the degree to which the data is representative of the real-world situation, completeness describes whether there is data missing or not, consistency shows how close the data aligns with another dataset or a reference dataset, uniqueness records if an event or object occurs multiple times, timeliness represents the time difference between the actual event time and the time the event is captured in a system and finally validity describes the closeness of data values to predetermined values or a calculation (Gawande, 2021).

Gawande (2021) also states that, although these six characteristics are commonly used, they are not universally agreed upon. Other characteristics they list in their article may be features like Currency, Conformity, Integrity, Precision, or several others.

As mentioned before, it is essential to determine what exactly the data should be used for. This can lead to a situation, where the timeliness characteristic can be more

important than, for instance, the uniqueness feature. A prime example of that would be data usage for a real-time application. While several of these characteristics are indirectly relevant to this thesis, the main emphasis lies in the completeness aspect.

2.2. Missing Data

Missing data is an obstacle that data scientists face on a regular basis. As described before, it is one of the main factors in terms of assessing the quality of the available data. The reasons for incomplete data are quite diverse according to Emmanuel et al. (2021), ranging from human errors when processing data, to machine errors due to malfunctions, to dropouts in studies, etc. The issues that arise from this problem can be for instance biased outcomes and mistakes in data analysis or performance degradation (Emmanuel et al., 2021).

A prominent example of significant problems resulting from missing data is the Challenger explosion from 1986. In January of that year, NASA launched the space shuttle Challenger with seven astronauts on board for its tenth mission. Just moments after the launch the space shuttle disintegrated in a fiery explosion, leading to the death of all crewmembers. (Uppenkamp, 2022).

Uppenkamp (2022) states that today, there is widespread agreement that the cause of the explosion was a sealing ring that failed under extremely low temperatures and high pressure during the launch. As a result, flammable gases escaped from their containers and ignited, ultimately leading to the disaster. Looking beyond the technical aspect, one finds that the seal ring problem was known well in advance and could have been prevented with better data management and decision-making procedures. In fact, concerns about the gasket were raised nine months earlier, and even the manufacturer objected to its use. Unfortunately, NASA management was operating on inconsistent and incomplete data. This resulted in the usage of different databases, which led to differing entries and ultimately to several gaps regarding the information about the aforementioned sealing rings (Uppenkamp, 2022).

Although this tragic accident cannot solely be blamed on an error in data management, according to Uppenkamp (2022) it shows the importance of complete and correct data and that even big organizations like NASA cannot be successful while ignoring this fact.

For the technical area of machine learning, incomplete data has its own challenges. Incomplete data is not ideal to train and test a machine learning model and even if the data is deleted or imputed, it can lead to biased results (Emmanuel et al., 2021).

2.3. Missingness Patterns for Data

Data can be missing for several reasons and in different patterns. In most of todays research regarding missing values in data, there are three types of missingness patterns considered:

- Data missing completely at random (MCAR)
- Data missing at random (MAR)
- Data missing not at random (MNAR)

2.3.1. Missing Completely at Random

As the name suggests, missing completely at random represents the highest level of random missingness. In these scenarios, the values that are absent are independent of the other values within the dataset and hold no connection to other absent values within their column as well, according to Rubin (1976).

This kind of missingness pattern is preferably used in most papers and studies, as it is usually the easiest missingness pattern to recreate. But in contradiction to the easiness of the recreation stands the probability of its occurrence. Realistic examples would be absent data because of bad luck or an unobserved data sample of members in the population, in which case each person has the same chance of being picked for this sample (van Buuren, 2018).

A simple example of this kind of missingness pattern can be seen in the following Figure 2.1, where the information regarding the height is absent completely at random.

Height	$\text{Height}_{\text{MCAR}}$
179.0	?
192.0	?
189.0	189.0
156.0	156.0
175.0	?
170.0	170.0
181.0	?
197.0	?
156.0	156.0
160.0	160.0

Figure 2.1.: MCAR Example (Jäger et al., 2021)

2.3.2. Missing at Random

Rubin (1976) states, that this kind of missingness pattern describes situations, where the missing data depends on values in other columns or features.

It is important to note that the absent values are independent of their respective column. Because of this, it is possible to detect a pattern that enables the Data Scientist to predict the missing values based on this pattern and the values in the other columns (García Laencina et al., 2010).

A typical example of this scenario would be a survey regarding personal information, where for instance women choose not to insert their weight or Lawyers do not insert their salary, while everyone with a different occupation does.

A further example of this kind of missingness pattern can be seen in the following Figure 2.2, where all male participants refuse to enter their height.

Height	Gender	$\text{Height}_{\text{MAR}}$
200.0	M	?
191.0	M	?
198.0	F	198.0
155.0	M	?
206.0	M	?
152.0	F	152.0
175.0	F	175.0
159.0	M	?
153.0	F	153.0
209.0	M	209.0

Figure 2.2.: MAR Example (Jäger et al., 2021)

2.3.3. Missing Not at Random

The last missingness pattern, which is defined by (Rubin, 1976) and generally used in research for data imputation, covers the remaining cases, where no randomness is involved. In this situation, the missing values are absent because of their actual value within their column.

This kind of missingness pattern is difficult to recognize since the reasons for the absent values can vary greatly. An example of this can be a weighting scale that wears out over time and produces more missing values along the way or a survey in which heavier people refuse to give an answer regarding their weight (van Buuren, 2018).

An example of this kind of missingness pattern can be seen in the following Figure 2.3, where the participants smaller than 181 cm refused to enter their height.

Height	$\text{Height}_{\text{MNAR}}$
154.0	?
181.0	181.0
207.0	207.0
194.0	194.0
153.0	?
156.0	?
198.0	198.0
185.0	185.0
155.0	?
164.0	?

Figure 2.3.: MNAR Example (Jäger et al., 2021)

2.4. Handling of Missing Data

As described in the previous chapter and the introduction, the data characteristic completeness is central to this thesis. Incomplete datasets with data missing for various reasons are a rather common sight for many data scientists. As stated by Emmanuel et al. (2021), it can have a serious impact on the results of an analysis or can affect the performance of a machine learning model, which is trained with incomplete data. In general, there are two possible ways to deal with these corrupted datasets.

2.4.1. Deletion of Data

The first option can be realized rather quickly, according to the article by 2U (2022), by simply deleting the data points which do not contain all the required information. Alternatively, they suggest the possibility to delete the affected features, depending on their importance and relevance to the required task. The negative side effects of this approach are that data is getting lost in the process and depending on the existing missing pattern in the dataset, it is possible to introduce a bias in the data, as described in the chapter about the missingness patterns (Jadhav et al., 2019).

2.4.2. Imputation of Data

The second possible approach is to impute the missing data (2U, 2022). Imputation describes the process of replacing missing values in the dataset with the most plausible possible value. There are multiple ways to determine these most plausible values, which will be introduced in more detail in the following sections. In general, there are statistical approaches that are usually more straightforward and require less computational power and there are approaches that are based on machine learning methods (Jadhav et al., 2019). In this thesis, a total of six different imputation methods are used. The theoretical background will be described in the following chapters while the exact usage in the experiments will be explained together with the Benchmark for Data Imputation Methods paper and Jenga Framework.

Mean/Mode

Mean and Mode are technically speaking two different approaches, but due to the fact that the used datasets contain numerical and categorical data, which also contain nominal data, a mixed approach is used here. The mean is calculated by the sum of all values divided by the number of values. For mode, the most frequent value is determined (Jadhav et al., 2019). This method serves mostly as a baseline and does not require a trained model for the implementation of the datasets. Jadhav et al. (2019) pointed out, that this yields a clear performance advantage in terms of computational costs, but can lead to a change in the shape of the distribution and highly biased parameter estimates.

K-NN (K-Nearest Neighbour)

According to the article "What is the k-nearest neighbors algorithm? by IBM" (2023) K-NN is a supervised machine learning method that can be used for regression and classification. It uses proximity between data points based on training data to make predictions or classifications for individual data points. To assign the fitting class label to the targeted point based on the closest neighboring points, K-NN utilizes methods like the Euclidean distance, the Manhattan distance, or the Hamming distance to determine the closest points. By defining the k with a number, the user can determine how many of the closest points should be taken into consideration ("What is the k-nearest neighbors algorithm? by IBM", 2023).

While it is easy to implement and requires only a few hyperparameters, the article "What is the k-nearest neighbors algorithm? by IBM" (2023) describes it as prone to overfitting and claims that it does not scale well in general, leading to significant increases in computational cost with increasing data sizes.

The article also states, that due to its characteristics, K-NN is most commonly used for simpler recommendation systems, data mining, pattern recognition, or similar activities.

Random Forest

The article "What is Random Forest? by IBM" (2023) describes Random Forest as a supervised machine learning algorithm that can be used for regression and classification tasks. This method combines the output of multiple decision trees, which are built on different samples, to get a single result. For a regression task, the individual decision trees will be averaged, while for a classification task, a majority vote (the most frequent categorical variable) will yield the predicted class ("What is Random Forest? by IBM", 2023).

The article "What is Random Forest? by IBM" (2023) states further, that due to its easy usage and flexibility, it has become quite popular and is now widely used for a multitude of challenges. Aside from those qualities, the Random Forest algorithm can be quite time-consuming, which should be taken into account, depending on the task at hand.

The authors describe also, that in general, it is often used for instance in e-commerce for recommendation systems or in finance in order to reduce time spent on data management and pre-processing tasks.

Discriminative Deep Learning

Machine learning models based on deep learning have increased significantly in popularity in the recent decade and their visualization as network has become almost synonymous with the term "artificial intelligence" for the general public. According to Reyes (2023), deep learning models are essentially neural networks with at least three layers. They were initially designed to imitate the behavior of the human brain. Neural networks consist of layers of nodes, which are inspired by the neurons in the brain. The nodes within the layers are connected to the nodes from the previous and following layers. In this artificial neural network signals travel between nodes and assign corresponding weights. A heavier-weighted node will exert more effect on the next layer of nodes. In the final layer, the weighted inputs will be compiled to produce an output (Reyes, 2023).

For this thesis, two different kinds of deep learning models are of interest, discriminative and generative, which will be explained in this and the following chapter. Turing (2022) states that discriminative machine learning models, which are also called conditional models, are often used for supervised machine learning. Their

main goal is to learn the boundaries and separate the data points into different classes by using probability estimates and maximum likelihood. This approach makes discriminative deep learning very suited for classification problems.

Generally speaking, deep learning has primarily been used for supervised learning models in the past. Artificial neural networks (ANN), recurrent neural networks (RNN), and convolutional neural networks (CNN) are examples of this kind of supervised learning model. These deep learning types are commonly used for computer vision tasks, like image recognition, and in more recent versions even in advanced fields like natural language processing or time series analysis (Turing, 2022).

Generative Deep Learning

Generative Deep Learning is the second approach to deep learning covered in this thesis. Turing (2022) describes, that in contrast to discriminative approaches, generative learning is classified as unsupervised machine learning. These models go in-depth to model the actual data distribution and learn the different data points, rather than modeling just the decision boundary between classes. Aside from the vulnerability to outliers, generative models have their own advantages and are especially suited for the prediction of probabilities since they focus on the probability distributions of the given data (Turing, 2022).

Variational Autoencoder

In this thesis two different generative deep learning approaches are used, the variational autoencoder (VAE) and the generative adversarial network (GAIN). Rocca (2021b) explains Autoencoders as neural network architectures composed of an encoder and a decoder that create a bottleneck the data must pass through. They are also trained to lose a minimal quantity of information during the encoding-decoding process. This can be achieved through training with gradient descent iterations with the target to minimize the reconstruction error (Rocca, 2021b).

He further states, that due to overfitting, the latent space of an autoencoder can be extremely irregular. This leads to the point, that makes it very difficult to define a generative process that simply consists to sample a point from the latent space and make it go through the decoder to get new data (Rocca, 2021b).

A variational autoencoder, according to Rocca (2021b), tackles the problem of latent space irregularity by making the encoder return a distribution over the latent space instead of a single point. Additionally, it adds a regularisation term over that returned distribution in the loss function to ensure a better organization of the

latent space.

By assuming a simple underlying probabilistic model to describe exemplary data, the intuitive loss function of VAEs, composed of a reconstruction term and a regularisation term, can be carefully derived, by using in particular the statistical technique of variational inference, which is also giving this particular autoencoder its name (Rocca, 2021b).

Generative Adversarial Networks

GAN models on the other hand are a bit less complex on a theoretical level, according to Rocca (2021a), which is why they have gained popularity in the scientific community. Like the VAE, generative adversarial networks belong to the set of generative models, which means that they are able to generate new data from a given, usually complex, probability distribution.

Rocca (2021a) further states, that therefore deep learning generative models are modeled as neural networks that take as input a simple random variable and that return a random variable that follows the targeted distribution.

By trying to trick another network, that is trained at the same time to distinguish generated data from original, true data, it is possible to indirectly train those generative adversarial networks. These kinds of networks are commonly used for instance in AI image generation (Rocca, 2021a).

3. Current State of Research

3.1. Research Methodology

For the literature research in this thesis, the author has employed an unsystematic approach. The starting point for this unsystematic search was the research paper "A Benchmark for Data Imputation Methods" from Jäger et al. (2021). Since this thesis is building on the work of Jäger et al. (2021), this approach makes sense and already covers several papers, which are relevant to this topic. By moving further and considering the sources of the sources from the previously mentioned paper from Jäger et al. (2021), more relevant content could be reviewed. In addition to that, the authors applied a systematic search via *Google Scholar* as well. Therefore he applied multiple keywords and combinations, as listed below, and filtered the results manually by actual relevance to this research topic.

- *Missing AND (Data OR Value) AND Imputation*
- *Missing Data Imputation Methods AND (Comparison OR Benchmark)*
- *Missing Data Imputation AND (Comparison OR Benchmark)*
- *Data Imputation Methods AND Comparison OR Benchmark*
- *Data Imputation*

While there was a focus on contemporary literature, the author did not limit the research to a certain time frame, although only english and german literature was taken into consideration.

3.2. Related Work

In the past years, several scientific papers were published, that focused on the handling of incomplete data in data science, specifically in the area of machine learning. This thesis is based in large part on the work of Jäger et al. (2021), and Schelter et al. (2021), which will be explained in detail in the following chapters. Prior to that, there was the need to evaluate and research the work of scientists

up to this point in time. Several researchers have investigated similar and related questions regarding data imputation and their work serves as a baseline for this thesis to build on.

3.2.1. Accuracy of Imputation

For a basic comparison of different methods for missing data imputation, the right selection of said methods is important. While the focus of this thesis is to evaluate which methods of imputation deliver a result best suited for training data in machine learning, there has been a scientific effort to examine how accurately single imputation methods can predict actual missing data values.

In their paper "Comparison of Performance of Data Imputation Methods for Numeric Dataset" Jadhav et al. (2019) took a closer look at the performance of several imputation methods performance. In total five datasets, ranging from 210 to 1030 data points, with 7 to 13 features were used as the basis for their experiments. To make a comparison the authors of the paper artificially corrupted the datasets by introducing a certain amount of missing values. The used proportions range from 10% up to 50%. For these imputation experiments, they used statistical approaches like mean, median, and sample, as well as machine learning-based approaches like K-NN, Bayesian Linear Regression, non-Bayesian Linear Regression, and Predictive Mean Matching.

Their findings indicate that K-NN usually outperforms all other used imputation methods, while Predictive Mean Matching comes in second best in most experiments.

While these results paint a rather clear picture, the limitations of the work of Jadhav et al. (2019) put them in perspective again. The very limited selection of datasets with relatively few data points does not inspire confidence for more common large-scale applications, especially considering the weakness of K-NN in regard to large-scale applications. Aside from that, the authors conducted this experiment without any hyperparameter optimization and only considered numeric datasets for their analysis, which is also an aspect that does not necessarily apply to most real-life use cases. Interestingly the authors discuss the different missingness patterns without stating which pattern they used in their own experiments. This vague approach again leaves room for interpretation.

The imputation research by Biessmann et al. (2018) explored the exact opposite direction by focusing on the imputation of missing, non-numeric data. Therefore they facilitated specifically deep neural networks, N-gram, and Long Short-Term Memory (LSTM), for their experiments. As a baseline for comparison, they also included mode and string matching. For their tests, the authors used two datasets with 10.000 and 333.106 data points as well as up to 100 features.

Their findings show that the deep learning approaches outperform the conventional methods significantly, although the differences between the two deep learning methods appear to be negligible.

Like the previously described paper by Jadhav et al. (2019), the limitations of this research by Biessmann et al. (2018) are quite significant to the core topic of this thesis. It is not clarified which kind of missingness patterns were considered, additionally, there is no clear description of the amount of data missing in the two datasets. Even though the size of the datasets makes this analysis very relatable to real-life applications, since only two datasets were used, the results may not be very representative of the actual performance of the deep learning approaches on a larger scale. This paper focused on pure imputation performance without considering the further usage of the imputed data for machine learning purposes.

3.2.2. Performance on Imputation and Downstream Machine Learning

Aside from the papers previously described, there is a section of research that goes beyond the simple comparison of imputed values to ground truth values. Poulos and Valle (2018) were investigating the differences in the downstream machine learning task performance between imputed and incomplete training data, with two binary classification datasets. To achieve this comparison, they corrupted their two datasets with two different missingness patterns, MNAR and MCAR, as well as with four different missingness fractions each, in this case, 10%, 20%, 30%, and 40%. Their datasets contained 435 and 48842 data points, as well as 14 and 16 features respectively.

It is also important to note, that the imputation itself was conducted solely on categorical data. For this, they used K-NN, mode, random replacement, logistic regression, random forest, and SVM (Support Vector Machine). After the imputation, each dataset was used to train three different machine learning models. For this task Poulos and Valle (2018) used decision trees, random forests, and artificial neural networks (ANN) as classification models. While the authors decided to optimize the hyperparameters for one of the three downstream machine learning tasks, they did not optimize them for any of the imputation models. As a reference, these models were trained on incomplete data as well. To make this possible, these perturbed datasets were one-hot-encoded before the training.

Poulos and Valle (2018) findings reveal that imputation certainly has the potential to improve the performance of downstream machine learning tasks. For the bigger dataset, the models trained on imputed training data always achieved better results than the models trained on incomplete data. K-NN performed the best in most scenarios, while the ANNs had a slight advantage towards the decision trees and random forest models. Most of the results were within 3% in terms of their

performance, which means there appears to be no drastic difference between the imputation methods. For the smaller dataset, the results varied a bit more. Also, the one-hot-encoded models achieved some competitive results. In summary, there is a tendency, that K-NN performs best in most situations.

Aljuaid and Sasi (2016) were conducting a rather similar experiment. They picked five datasets, ranging from 150 to 30162 data points and up to 16 features, perturbed them in advance, and imputed the data again. The amount of perturbed data lay between 10% and 25%, while they used the three missingness patterns MAR, MCAR, and MNAR. Unlike Poulos and Valle (2018), Aljuaid and Sasi (2016) did not use the single missingness fractions and missingness pattern for all the available datasets. Instead, they used one particular missingness fraction and one particular missingness pattern per dataset. After the datasets were imputed with one of the five methods (mean/mode, K-NN, expectation maximization, hot-deck imputation, and C5.0 decision tree), a machine learning model for classification was trained with the now imputed dataset. As a reference, the same classification model was trained with the original, complete dataset as well.

In their findings, Aljuaid and Sasi (2016) concluded that hot deck imputation delivers the best results. A closer look at the exact numbers however reveals that the margins in the overall performance were rather small.

While both these papers have a similar direction as this thesis, there are significant limitations to them, which curtail their transferability and universality. Two and five datasets, as well as the rather low amount of data points on several of them, do not necessarily give a good impression of the overall effectiveness of data imputation for the data pre-processing in machine learning. While Poulos and Valle (2018) tried to ensure a certain level of comparability due to the application of the same missingness patterns and missingness fractions to all of the datasets and experiments, this comparability is not as predominant in the work of Aljuaid and Sasi (2016). Aside from that, they limited themselves exclusively to classification downstream machine learning tasks.

3.2.3. Performance on Imputation and Downstream Machine Learning without Ground Truth

While the authors of the previously described papers have mostly worked with comparisons to the ground truth, the scientific researchers Woźnica and Biecek (2020) have focused their studies on imputation without available ground truth. Therefore, they evaluated and compared seven different imputation methods in combination with five classification models regarding their predictive performance.

The authors selected for their experiment a total of 13 datasets, which are exclusively for binary classification tasks. The datasets contained between 57 and 48842 data

points and ranged between 10 and 123 features. As mentioned before, Woźnica and Biecek (2020) choose specific datasets, which were missing values in at least one column from the beginning, therefore working without ground truth for the imputation. As a consequence, every dataset was comprised of a different, often distinct, missingness fraction. The respective missingness pattern was also unknown. For the imputation, Woźnica and Biecek (2020) used mean, random, softImpute, missForest, VIM K-NN, VIM hotdeck, and MICE. For the downstream machine learning models, they used logistic regression, classification tree, Random Forest, K-NN, and XGBoost.

Woźnica and Biecek (2020) imputed the datasets with each imputation method and trained with each imputed dataset five machine learning models, which used the five previously mentioned approaches respectively. Afterwards, the results were compared and ranked.

Although the results do not clearly identify the best imputation method, Woźnica and Biecek (2020) were able to answer their initial question, of whether imputation matters or not. From the findings of the experiment, it is clear, that the combination of the usage of imputation methods and subsequent downstream machine learning models influences the results of said downstream model.

3.2.4. Benchmarks for Developed Imputation Frameworks

The following two scientific papers differ from the other described articles. Zhang et al. (2018) and Bertsimas et al. (2017) introduced their own method, which they compared against existing approaches. While the aim was to showcase the improvements of their introduced methods, due to the transparency in terms of the published results, it is possible to compare the results for the existing approaches amongst each other, at least to a certain degree.

Zhang et al. (2018) introduced an iterative expectation-maximization (EM) algorithm that learns and optimizes a latent representation of the data distribution, parameterized by a deep neural network, to perform the imputation.

Zhang et al. (2018) used 13 datasets, ten for classification and three for regression for their experiments. Those datasets ranged from 86 to 2600 data points with up to 60 features. These datasets were perturbed with three missingness fractions of 25%, 50%, and 75%, although only MNAR was used as a missingness pattern. The datasets were afterward imputed with in total 11 baseline methods (zero, mean, median, MICE, miss-Forest, softImpute, K-NN, PCA, autoencoder, denoising autoencoder, and residual autoencoder) in addition to the implemented solution by the authors.

In their findings, Zhang et al. (2018) concluded, that their proposed framework

outperformed the baseline methods in most scenarios. The comparison between the baseline methods themselves revealed no clear favorite.

Aside from Jäger et al. (2021), Bertsimas et al. (2017) provided the largest and most extensive comparison of the papers that were reviewed for this thesis. For their research, the authors introduced a flexible framework based on formal optimization to impute missing data with mixed continuous and categorical variables. This framework can readily incorporate various predictive models including K-NN, support vector machines, and decision tree-based methods, and can be adapted for multiple imputations. The algorithm cross-validates the choice of the best imputation method out of the previously mentioned imputation methods, where the hyperparameters are also cross-validated.

For this research study Bertsimas et al. (2017) used in total 84 datasets, which were ranging from 23 to 5875 data points and had up to 124 features. Regarding the missingness pattern, they used MNAR and MCAR. The introduced missingness fractions for the datasets ranged from 10% up to 50%.

The experiment can be split into two parts. In the first part, the datasets were artificially perturbed as described and afterward imputed with five different imputation methods (mean, K-NN, iterative K-NN, predictive mean matching, Bayesian PCA) as well as with the proposed framework. After that, Bertsimas et al. (2017) compared the results to the ground truth and determined the best single imputation method per dataset.

For the second part, Bertsimas et al. (2017) selected 10 datasets, five for regression and five for classification. Then they trained classification and regression machine learning models with those datasets, after they were imputed. For this training, they used the best single imputation method per dataset, as well as their method and MICE as a multiple imputation approach. The classification was realized through SVM and optimal trees and the regression was handled through LASSO and SVR models. It is important to note, that the training test split was 50/50 in their experiments, which could have impacted the results. Afterward, they compared the performance of the machine learning models.

Bertsimas et al. (2017) findings concluded, that their proposed framework outperformed the other methods on average by 8,3% for the mean absolute error (MAE). The full numeric results showed that this was closely followed by K-NN and Bayesian PCA. The results also suggested that single imputation approaches suffer in performance for very small datasets.

To summarize the findings in the current literature on this topic, there are various studies, which have investigated different aspects of missing data imputation. Several papers only focus on how accurately imputation methods can predict the ground truth, while some papers went further and investigated the impact on downstream

machine learning models. Although most of these studies reveal certain conclusions, most of them are unable to provide a clear favorite imputation method for either research focus. Nevertheless, they were able to show, that simpler imputation methods cannot be ruled out as they often achieve competitive results and therefore deserve further consideration for this thesis. Most authors limited their experiments in scale, which lead to the situation, that either very few or very small datasets were used for their tests, and in some cases both. These small and few datasets are certainly easier to manage and require less computational cost, but this also severely limits the validity of their results and their transferability to real-world applications. In addition to this, a competitive, large-scale comparison of different imputation methods and their impact on downstream machine learning models as a central focus was lacking in the literature reviewed for this thesis.

All of these limitations illustrate the need for a further, competitive comparison of different imputation methods and especially their impact if used as training data for machine learning models.

4. A Benchmark for Data Imputation Methods and Jenga

This thesis is intended to build on the work of Jäger et al. (2021) and Schelter et al. (2021). Schelter et al. (2021) proposed in their scientific paper a framework with the name Jenga. This framework is meant to help study the impact of data errors on the predictions of machine learning models. Jäger et al. (2021) were building their data imputation experiments on the basis of that framework. Therefore it is important to outline the content of their work as well as their limitations.

4.1. Jenga

4.1.1. Motivation and Target of the Paper

Machine learning practitioners and data scientists require a lot of time to deal with a diverse set of problems regarding the quality of their working data. In many cases, it is also quite difficult to anticipate what consequences certain approaches have, when applying them to solve the present issues with data quality. Especially when data is used to train machine learning models, the quality of the models can suffer severely. In order to deal with that problem, Schelter et al. (2021) propose this framework, a lightweight, open-source experimentation library to study the impact, as well as mitigation techniques for data errors in machine learning models. Jenga enables users to test the robustness of their models against frequent data errors. It includes an abstraction for prediction tasks based on a dataset and a model, an easily extendable set of synthetic data corruptions (for example, for missing values, outliers, or noisy measurements), as well as an evaluation function to experiment with different data corruptions methods.

4.1.2. Framework Design

The Jenga framework is designed around three core abstractions:

i) tasks contain a raw dataset, a machine learning model, and represent a prediction task

ii) data corruptions take raw input data and randomly apply certain data errors to them (e.g., missing values)

iii) evaluators take a task and data corruptions and execute the evaluation by repeatedly corrupting the test data of the task and recording the predictive performance of the model on the corrupted test data (Schelter et al., 2021)

The Jenga framework provides several types of data corruption. It is possible to swap values, wherein a specific ratio of values in one column is replaced with values from another column. Other typical corruptions include improper scaling of numerical values or the introduction of noise. In this case, a fraction of the data is perturbed by adding Gaussian noise, centered at the data point with a standard deviation randomly selected from the interval of two to five. It represents a classical error in measurement data. Encoding errors, where certain characters in string attributes are replaced, or image corruptions are also within the scope of this framework.

The most interesting data corruption feature for this thesis is the introduction of missing data. The Jenga framework does not only provide the user with the opportunity to introduce a predefined amount of missing data, it also enables the usage of different missingness patterns, as described in the previous section 2.3.

The second major part of the framework contains the evaluation and impact of these data errors on the predictive performance of machine learning models. Therefore Jenga provides two different evaluators. The ***Corruption-Impact-Evaluator*** requires a given task, a trained model, and a manually chosen list of corruptions. It applies each data corruption to the tasks' held-out test set and computes the models' predictive performance considering the corruption. The second evaluator enables users to additionally integrate a data validation schema of their choice into the evaluation.

4.2. Benchmark for Data Imputation Methods

4.2.1. Motivation and Target of the Paper

In their scientific paper, Jäger et al. (2021) build their experimental settings around the previously described Jenga framework. While there are several research papers about the issue of handling missing data via imputation, comprehensive benchmarks comparing classical and modern imputation approaches under fair and realistic conditions are apparently underrepresented. Jäger et al. (2021) tried to fill this gap with their study, by conducting a comprehensive set of experiments on a larger number of datasets with realistic missing conditions and six different imputation approaches. Each imputation method is evaluated in terms of imputation quality and the impact of the imputation on a downstream machine learning task.

4.2.2. Experiment Preparation

Prior to the experiments Jäger et al. (2021) defined multiple settings. Most important were the selection of the used datasets, the selection of the imputation methods, and the evaluation metrics for the results of their experiments.

Datasets

In total 69 datasets were used by Jäger et al. (2021) for these experiments. These datasets contained numerical, as well as categorical data and covered three different kinds of tasks for the downstream machine learning experiments:

- 21 regression datasets
- 31 binary classification datasets
- 17 multiclass classification datasets

These datasets were accessed via an OpenML API during the experiments. Since Jäger et al. (2021) ran in total more than 4900 tests on these datasets, it was necessary to limit the number of instances per dataset between 3000 and 100000, as well as the number of features between 5 and 25. Additionally, they removed any duplicates, Sparse ARFF formatted datasets, and any corrupted datasets.

Imputation Methods

The six imputation methods chosen for these experiments are listed as follows:

- Mean/Mode
- K-NN
- Random Forest
- Discriminative Deep Learning
- Variational Autoencoder
- Generative Adversarial Network (Jäger et al., 2021)

The theoretical background to each method is described in subsection 2.4.2. The implementation of these approaches will be described in more detail in this chapter. For mean and mode imputation there is not much to add since it is a straightforward statistical approach, however, more can be said about the exact methods of the other imputation techniques.

For K-NN imputation Jäger et al. (2021) utilized the so-called Hot-Deck imputation, as described in Batista and Monard (2003). In practice, Jäger et al. (2021) facilitated the KNeighbors Classifier for categorical to-be-imputed columns and KNeighborsRegressor for numerical columns, which are provided by scikit-learn. In terms of hyperparameter optimization, the authors used 5-fold cross-validation and a grid size consisting of three values (1, 3, 5) for *n_neighbors* as shown in Listing 4.1.

```
1     "knn": {
2         "hyperparameter_grid_categorical_imputer": {
3             "n_neighbors": [1, 3, 5]
4         },
5         "hyperparameter_grid_numerical_imputer": {
6             "n_neighbors": [1, 3, 5]
7         }
8     },
```

Listing 4.1: K-NN Hyperparameter Grid

Random Forest imputation is handled rather similar to the K-NN imputation. In this case, Jäger et al. (2021) used the packages provided by scikit-learn as well, RandomForestClassifier and RandomForestRegressor for categorical and numerical columns respectively. Like in the case of K-NN, the authors used 5-fold cross-validation and grid size with three values (10, 50, 100) to optimize the hyperparameter *n_estimators* as shown in Listing 4.2.

```
    "forest": {
        "hyperparameter_grid_categorical_imputer": {
            "n_estimators": [10, 50, 100]
        },
        "hyperparameter_grid_numerical_imputer": {
            "n_estimators": [10, 50, 100]
        }
    },
```

Listing 4.2: Random Forest Hyperparameter Grid

The discriminative deep learning imputation technique was implemented using the AutoML3 library autokeras (Jin et al., 2019). The authors utilized StructuredDataClassifier from autokeras for categorical columns and StructuredDataRegressor for numerical columns. Both mentioned modules use their respective default hyperparameter, if not defined otherwise. Most notably the default loss functions for the StructuredDataRegressor facilitates the *mean_squared_error* (Jin et al., 2023), while the StructuredDataClassifier applies either *binary_crossentropy* or *categorical_crossentropy* for the loss function, depending on the number of classes (Jin et al., 2023-01). The data was properly encoded by both classes, and they also optimized the models' architecture and hyperparameters as shown in Listing 4.3. In order to train each model for a maximum of 50 epochs (Autokeras applies early stopping by default), Jäger et al. (2021) set the parameters *max_trials=50* and *epochs=50*. This instructs Autokeras to try up to 50 different model architectures and hyperparameter combinations.

```
    "dl": {
        "max_trials": 50,
        "tuner": None,
        "validation_split": 0.2,
        "epochs": 50
    },
```

Listing 4.3: Discriminative Deep Learning Hyperparameter Grid

The Variational Autoencoder Imputation method Jäger et al. (2021) focused on the original VAE imputation approach in order to ensure comparability. The authors used the method suggested by Camino et al. (2019) to determine the best model architecture, i.e., the number and size of hidden layers. For the encoder and decoder, they optimized using zero, one, or two hidden layers and fixed their sizes in relation to the number of columns in the table, also called the input dimension. If present, the first hidden layer of the encoder contained 50% of the neurons from the input layer and the second layer, 30%. For the upsampling of the data to the same size as the input data, the decoder sizes were the exact opposite. Additionally, the latent

space was set at 20% of the input dimension. Jäger et al. (2021) used the Adam optimizer with the default hyperparameters like a batch size of 64, as well as early stopping within 50 epochs. For the encoder and decoder the activation functions *relu* (Rectified Linear Unit) and *sigmoid* were used. To optimize the n_hidden_layers hyperparameter the authors facilitated a grid size of three values (0, 1, 2), shown in Listing 4.4, but in contrast to the previous two imputation approaches, only a 3-fold cross-validation was used. The authors argued that this is required to reduce the overall training time.

```
    "vae": {
        "hyperparameter_grid": {
            "neural_architecture": {
                "latent_dim_rel_size": [0.2],
                "n_layers": [0, 1, 2],
                "layer_1_rel_size": [0.5],
                "layer_2_rel_size": [0.3],
            }
        }
    }
```

Listing 4.4: VAE Hyperparameter Grid

The last imputation method used in this scientific paper is the GAN-based imputation approach. Therefore, the authors utilized one of the most popular approaches of GAN-based imputation, Generative Adversarial Imputation Nets (GAIN) (Yoon et al., 2018).

```
    "gain": {
        "hyperparameter_grid": {
            "gain": {
                "alpha": [1, 10],
                "hint_rate": [0.7, 0.9]
            },
            "generator": {
                "learning_rate": [0.0001, 0.0005],
            },
            "discriminator": {
                "learning_rate": [0.00001, 0.00005],
            }
        }
    },
```

Listing 4.5: GAIN Hyperparameter Grid

GAIN adapts the original GAN architecture as follows. The input to the generator is the concatenation of the input data and a binary matrix representing the missing

values. The discriminator learns to reconstruct the mask matrix. Its input is the concatenation of the output of the generator and a hint matrix that reveals partial information about the missingness of the original data. The computation of the hint matrix includes the introduced hyperparameter *hint_rate*. To balance the generator's performance for observed and missing values, GAIN adds a second hyperparameter *alpha*. With the exception of the learning rate for the generator and discriminator, batch size of 64, and early stopping after 50 epochs, the authors used the Adam optimizer for training. To optimize this imputation approach, Jäger et al. (2021) facilitated a size 16 grid, again with a 3-fold cross-validation to reduce the training time, with two values for each of the following 4 hyperparameters, also to be seen in Listing 4.5.

Evaluation Metrics

To evaluate their experiments, Jäger et al. (2021) used two metrics: root mean square error (RMSE) and macro F1-score. The RMSE is defined as follows:

$$\text{RMSE} = \sqrt{\frac{\sum_{i=0}^{N-1}(y_i - \hat{y}_i)^2}{N}} \tag{4.1}$$

where N is the number of observations, y_i are the observations, and $\hat{y}_i$ are the predictions. The macro F1-score on the other hand is defined as the mean of the class-wise F1-scores:

$$macroF1 = \frac{1}{C}\Sigma_{i=0}^{C} F1_i \tag{4.2}$$

where i is the class index, C is the number of classes, and F1 is defined as follows:

$$F1 = \frac{TP}{TP + \frac{1}{2} * (FP + FN)} \tag{4.3}$$

where TP equals True Positive Count, FP equals False Positive Count, and FN equals False Negative Count. The F1 score is used to define the performance of classification tasks while the RMSE is used for the regression tasks. While for the latter a small value indicates a better performance, for F1 scores a higher value is preferred.

4.2.3. Experiments

Jäger et al. (2021) were conducting two experiments, each with two different scenarios. The first experiment was aiming to determine how accurately the imputation methods can impute the original values, while the second experiment should show the impact of the different imputation approaches on numerous downstream machine learning tasks. The first scenario is covering situations in which the authors use the original, complete datasets for the training of the model, the second scenario covers the training of a model on incomplete data.

Data Preparation

Jäger et al. (2021) defined a framework that provided a common API with the methods *fit* and *transform* for each of the six implemented imputation approaches. *Fit* trains the imputation model on given data while cross-validating a set of hyperparameters, while *transform* allows the imputation of the missing values of the to-be-imputed column the imputation model is trained on.

Two previously mentioned functionalities are offered by the Python module jenga (Schelter et al., 2021), which were utilized to carry out the experiments. The techniques to discard data for the selected fraction of data that should be corrupted and the missingness patterns MCAR, MAR, and MNAR were first put into practice. The patterns are explained in detail in section 2.3. Second, it establishes an 80/20 training-test split and acts as a wrapper for OpenML datasets, and can automatically develop a base model for the downstream task specified by the dataset.

Jäger et al. (2021) used the default task settings provided by Jenga in which scikit-learn's SGDClassifier and SGDRegressor are used for classification and regression tasks respectively. As part of the preprocessing, it first replaces missing values with a constant, then one-hot encodes categorical columns and normalizes numerical columns to zero mean and unit variance. Finally, it uses grid search to perform a 5-fold cross-validation of the hyperparameters *alpha*, *loss*, and *penalty* in order to train a robust model. The hyperparameters can be seen in the following exemplary Listing 4.6 for the binary classification tasks.

```
    param_grid = {
        'learner__loss': ['log'],
        'learner__penalty': ['l2'],
        'learner__alpha': [0.00001, 0.0001, 0.001, 0.01]
    }

    pipeline = Pipeline(
        [
            ('features', feature_transformation),
            ('learner', SGDClassifier(max_iter=1000, n_jobs=-1))
        ]
    )
```

Listing 4.6: SGDClassifier Hyperparameter Grid

Jenga reports the performance of the baseline model on the test set afterward (F1 for classification, RMSE for regression).

Considering the amount of time it took to conduct all experiments and to cut down on the required computational resources, Jäger et al. (2021) decided to manually sample one to-be-imputed column per dataset upfront, which remained static for all their experiments.

Experiment 1 – Imputation Quality

The goal of Jäger et al. (2021) was to demonstrate the accuracy of the imputation methods by performing this experiment. They used the Jenga framework to distribute the desired number of missing values among all the test set's columns. For a given missingness pattern and fraction, e.g., 30% MAR, they inserted $\frac{30\%}{N}$ missing values of that pattern into each of the N columns. The imputation quality evaluation was then performed by comparing the discarded values of the to-be-imputed column as ground truth and the predictions of the imputation model. If the to-be-imputed column was categorical, the F1s were reported. For numerical columns, the RMSE was reported.

Jäger et al. (2021) were focusing primarily on point estimates of imputed values. Therefore the assessment of the inherent uncertainty of imputed values was not taken into consideration for their work.

Experiment 2 – Impact on Downstream Machine Learning Tasks

In the second experiment, Jäger et al. (2021) tried to evaluate the impact of the different imputation approaches on several downstream machine learning tasks.

Due to the fact, that it was necessary for the discriminative model approaches to train one imputation model for each column with missing values, as well as due to the large number of experimental conditions, Jäger et al. (2021) decided to limit the scope of their work further in order to cut down on the vast computational costs. As a result, they decided to discard values only in the test sets' to-be-imputed column.

The entire experiment can be summarized in three steps. First, Jäger et al. (2021) trained the baseline model of the downstream machine learning task on the training set and reported its baseline score on the uncorrupted test set. After discarding values in the to-be-imputed column, the trained baseline model was used again. After that they calculated its score on the incomplete test set, hence the name incomplete. They then imputed the missing values of the test set and were using the trained baseline model on it. Finally, the authors calculated the imputed score.

In the end, they reported the impact on the downstream task's performance as the percent change of the imputation (*imputed*) over the incomplete data (*incomplete*) relative to the baseline performance on fully observed test data (*baseline*), as seen in the following equation.

$$impact\,on\,downstream\,task = \frac{imputed - incomplete}{baseline} \tag{4.4}$$

Scenario 1 – Training on Complete Data

For the previously described experiments, Jäger et al. (2021) added two different scenarios, which aim to cover common situations for machine learning researchers or data scientists. The first scenario covers model training with complete data.

Therefore the authors use the original data from the complete dataset to train the imputation model, which is then used for both experiments described in the previous chapter.

Scenario 2 – Training on Incomplete Data

Another common scenario Jäger et al. (2021) wanted to cover is the situation in which not only the test data but also the training data is corrupted. Therefore the experiments were slightly adjusted, by also corrupting the training data before the training of the baseline model.

Since training on incomplete data can lead to problems, the authors preprocessed the training data before the actual training. For discriminative imputation approaches, they replaced missing values with their column-wise mean/mode value. Categorical

columns were one-hot coded, and the data was normalized to zero mean and unit variance, as depicted in the following Listing 4.7.

```
1     categorical_preprocessing = Pipeline(
2         [
3             ('mark_missing',
                 SimpleImputer(strategy='most_frequent')),
4             ('encode', self._encoder)
5         ]
6     )
7     numeric_preprocessing = Pipeline(
8         [
9             ('mark_missing', SimpleImputer(strategy='mean')),
10        ]
11    )
```

Listing 4.7: Temporary SimpleImputer Settings for Scenario 2

In the case of generative imputation approaches, Jäger et al. (2021) had to preserve the number of columns. Therefore, they encoded the categories of categorical columns as values from 0 to n-1 where n was the number of categories. Then, the missing values were replaced with random uniform noise from 0 to 0.01, and finally, the data was min-max scaled from 0 to 1.

Additionally, due to the lack of a real baseline, Jäger et al. (2021) needed to adjust the calculation for the impact on the downstream machine learning task, by replacing the baseline denominator with *incomplete*. Therefore they reported the percent change of the imputation over the incomplete data relative to the downstream task performance on incomplete data.

$$\textit{impact on downstream task} = \frac{\textit{imputed} - \textit{incomplete}}{\textit{incomplete}} \tag{4.5}$$

4.2.4. Results and Findings

In order to evaluate the results, Jäger et al. (2021) used a ranking system to determine the best imputation method. Therefore, they facilitated the F1 scores and the RMSE and ranked the six imputation methods for each experiment, consisting of one distinct missingness pattern and one missingness fraction of data for one dataset. The imputation approach, which performed the best, was ranked 1 and the worst was ranked 6. If the model training failed for one imputation method, then that method was automatically assigned the worst rank.

Experiment 1 - Imputation Quality

The findings of Jäger et al. (2021) show that Random Forest, K-NN, and Discriminative Deep Learning performed the best in most conditions. Generally, the Random Forest Imputation approach tended to perform the best by achieving the best rank more often than the other imputation methods. Generative Deep Learning on the other hand had a tendency to perform the worst.

For the GAIN approach, this result was aided by the fact, that in about 33% of cases, the model training failed, which lead to the worst rank on those occasions. An investigation of these errors by the authors showed that GAIN's discriminator loss eventually went NAN, which lead to failures in further calculations and a failed training process. This depended strongly on the learning rate of the discriminator and the dataset. GAN-based models are generally known to be hard to train, which is why improvements for training GANs were being introduced to make their training process more robust, examples include the work of Salimans et al. (2016); Heusel et al. (2017); and Miyato et al. (2018). Since Jäger et al. (2021) declared, that a further hyperparameter optimization per dataset would have been out of the scope of their work, they decided to define the hyperparameter grids once and incorporated the imputation methods robustness regarding their hyperparameters into their evaluation.

A rather surprising result was the competitive performance of the mean/mode approach for the most challenging MNAR experiments with a high fraction of missing data.

Similar to training on fully observed data, K-NN, and discriminative deep learning outperformed generative deep learning methods in most settings with the model training on incomplete data. In the MNAR conditions, the imputation quality of all imputation approaches degraded in favor of mean/mode, which outperformed the others for 30% and 50% missingness fractions.

Experiment 2 - Impact on the Downstream Task

The results of this experiment aimed to answer the question, of whether imputation on incomplete test data improves the predictive performance of a downstream machine learning model.

The downstream performance was compared to the performance score obtained on incomplete test data, normalized by the machine learning model performance on fully observed test data. Overall, the classical machine learning methods and Discriminative Deep Learning performed best, achieving relative improvements of up to 10% and more, relative to fully observed test data.

Regarding the impact on the downstream tasks of the six imputation methods trained on incomplete data, it can be observed, that in regression tasks there were no considerable improvements. For the classification, Jäger et al. (2021) were able to document a slight improvement in a few settings, but also some significant negative effects for higher missingness fractions.

Generally, the results of the experiments show, that imputation can have a substantial positive impact on predictive performance in downstream machine learning tasks. While Jäger et al. (2021) were not able to identify a single best imputation method to achieve superior results, they were able to show that in more than 75% of their experiments, the predictive performance can be increased by at least 10%. Considering the notable differences in wall-clock run time, the authors suggested that random forest imputation delivered the best results considering its performance improvement and overall computational time. These improvements are especially notable for imputation models that were trained on fully observed data. The positive impact of imputing missing values in the test data was much smaller, sometimes even negative, when the imputation methods were trained on incomplete data.

5. Methodology

For this thesis, the author applied the so-called Design Science Methodology. According to Brocke et al. (2020), Design Science Research is a problem-solving paradigm that seeks to advance human knowledge through the creation of innovative artifacts, represented by constructs, models, methods, and instantiations (Gregor & Hevner, 2013; Hevner et al., 2004).

Put simply, Design Science Research seeks to advance the knowledge base of technology and science through the creation of innovative artifacts that solve problems and improve the environment in which they are instantiated.

Brocke et al. (2020) explains further, that the Design Science approach includes in total six steps, which describe the process applied in practice:

- Activity 1: Problem identification and motivation
- Activity 2: Define the objectives for a solution
- Activity 3: Design and development of an artifact to solve the problem
- Activity 4: Demonstration of the problem-solving ability of the artifact
- Activity 5: Evaluation of the demonstration
- Activity 6: Communication of all relevant information to the stakeholders

6. Implementation and Experiments

This chapter outlines the experiments conducted for this thesis.

6.1. Limitations of Current Literature

As reviewed in chapter 3, the currently available literature does, to the best of the authors' knowledge, not provide a competitive, large-scale comparison for the impact of missing data imputation on the predictive performance of downstream machine learning models. Therefore, this thesis aims to fill this gap by benchmarking the six aforementioned imputation methods against each other. Furthermore, based on the results of these benchmark experiments, there is an additional test considered, in which the decision, of which imputation method a practitioner should use for the dataset at hand, will be automated.

6.2. Imputation Experiments

While Jäger et al. (2021) were focussing primarily on the imputation for the test data, this thesis takes a different approach towards investigating the impact of missing data imputation. In addition to the test data, these experiments are expanded to the training data as well. Therefore the six imputation methods (mean/mode, random forest, K-NN, discriminative deep learning, VAE, and GAIN) are also used on the previously perturbed training data, which in turn will be used to train the downstream machine learning model. The following Figure 6.1 illustrates the adjusted experiment process for these experiments.

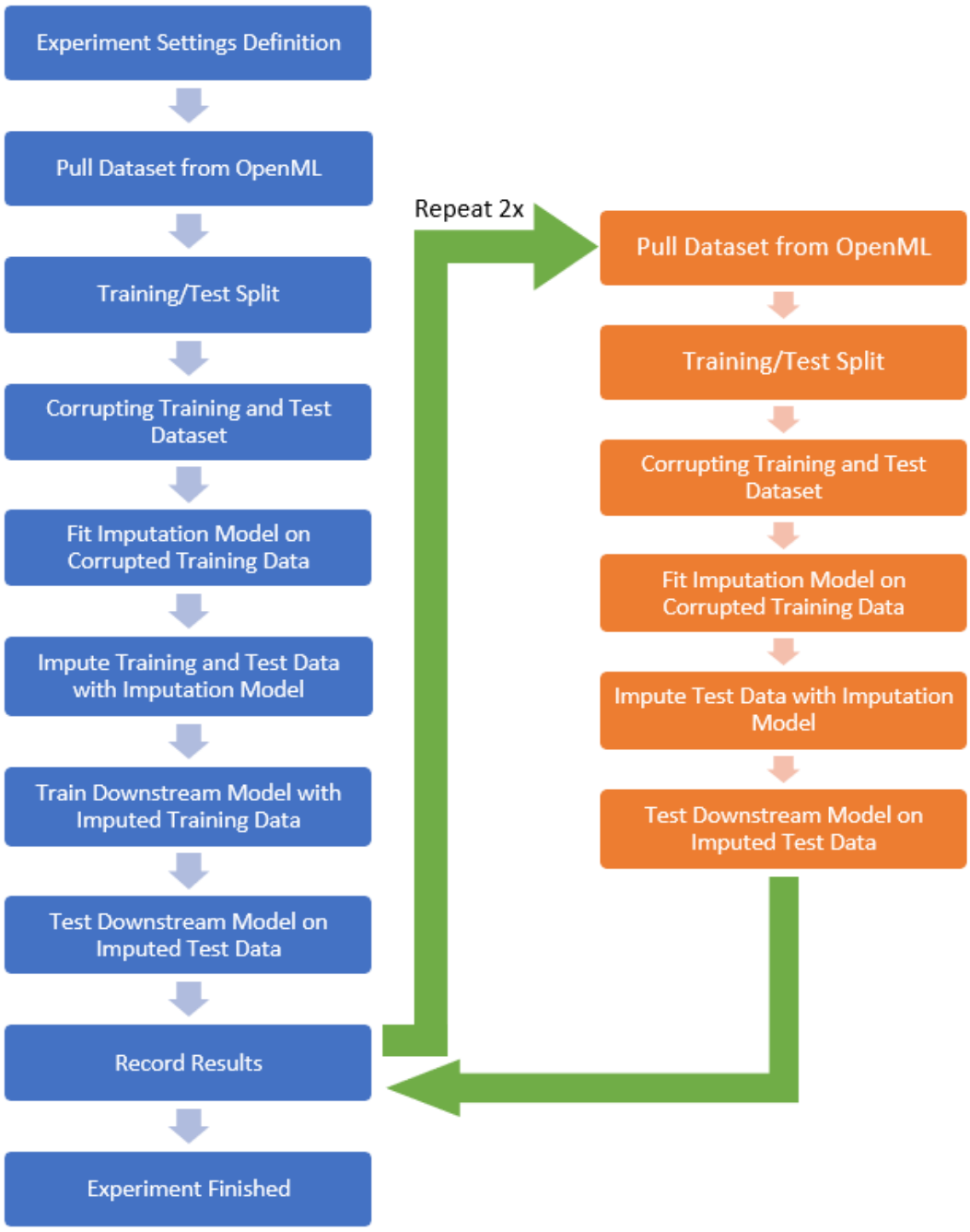

Figure 6.1.: Experiment Process

For the first step, the user must define the parameters of the experiment manually. This consists of the following information:

- Dataset ID
- Imputation Method
- Experiment Name
- Missing Fractions
- Missingness Patterns
- Strategies
- Number of Repetitions per Experiment
- Path to Result Storage Folder

The Dataset ID and Imputation Method are variable for the experiments conducted for this thesis. Based on the selected datasets, whose selection is described in detail in chapter 4, and imputation methods, which are described in detail in chapter 2 (theoretical background) and chapter 4 (application within the experiment), the remaining parameters remain static for all experiments. For missingness patterns and missing fractions, this thesis closely follows the parameters utilized by Jäger et al. (2021) and uses MCAR, MAR, and MNAR for the missingness patterns, as well as 1%, 10%, 30%, and 50% for the missing fractions. Jäger et al. (2021) were experimenting with different imputation strategies, but for this thesis, the focus lies on single-column imputation. In order to solidify the results, the experiments are conducted three times each. It is important to state, that one separate experiment consists of one dataset, one imputation method, one missing fraction, and one missingness pattern each. If the user defines for instance more than one missingness pattern at the start, then there will be multiple experiments for this one command. An example of this can be seen in the following Listing 6.1.

```
1 python run-experiment.py 42545 knn corrupted_experiment_reg
    --missing-fractions 0.5 --missing-types MAR --strategies
    single_single --num-repetitions 3 --base-path ../results
```

Listing 6.1: Example Command to Start Experiment

After the settings are clear, the program facilitates the OpenML API to pull the requested dataset from OpenML and conducts the training/test split for the dataset. For these experiments, a split of 80% training data and 20% test data is used. In order to ensure reproducibility for future practitioners, as well as comparability between the single experiments, the random seed for that training/test split remains static throughout the experiments.

After the dataset is pulled and split, training and test data are getting corrupted according to the experiment settings defined by the user. For this step, the Jenga functionalities are utilized (Schelter et al., 2021). As previously mentioned, every experiment is conducted three times. While the training/test split is identical for every repetition, the specific data points, that are perturbed, are different from repetition to repetition. To ensure that the results are deterministic and comparable, the random seed for each repetition is set at the beginning of the experiment. Aside from that, the data is only corrupted in the predefined, static column for each dataset.

Now that the training and test sets are corrupted, the next step is to set up the imputation model. Therefore the model is fitted with the corrupted training data. In addition, for further analysis, a timer is set to calculate the time it takes to fit the imputation model. With the imputation model fitted, the program moves on to utilize said imputation model to impute the corrupted training and test data.

After the imputation of the corrupted datasets, the program is using the imputed training data to fit the downstream machine learning model for the experiment. Therefore, the process is rather similar to the approach from Jäger et al. (2021). The data gets preprocessed with scikit-learn's OneHot-Encoder for categorical data and scikit-learn's StandardScaler for numerical data. Afterward, the downstream machine learning model is trained. For that, the program facilitates scikit-learn's SGDClassifier for classification tasks and scikit-learn's SGDRegressor for regression tasks. The author uses the default settings defined in the Jenga framework.

The SGDClassifier uses logistic regression, a probabilistic classifier ("sklearn SGDClassifier", 2023) as loss function Listing 4.6, while the SGDRegressor covers *squared_loss*, which refers to the ordinary least squares fit ("sklearn SGDRegressor", 2023), and *huber*, which modifies *squared_error* to focus less on getting outliers correct by switching from squared to linear loss past a distance of epsilon ("sklearn SGDRegressor", 2023), in its parameter grid, as depicted in Listing M.8.

For comparability, the hyperparameters are the same as Jäger et al. (2021) used for their experiments. To ensure further reproducibility, the random seed for the downstream machine learning models remains also static throughout the experiments.

```
        param_grid = {
            'learner__loss': ['squared_loss', 'huber'],
            'learner__penalty': ['l2'],
            'learner__alpha': [0.00001, 0.0001, 0.001, 0.01]
        }

        pipeline = Pipeline(
            [
                ('features', feature_transformation),
                ('learner', SGDRegressor(max_iter=1000))
            ]
        )
```

Listing 6.2: SGDRegressor Hyperparameter Grid

Via scikit-learn's GridSearch module, the program is selecting the best estimators for the final model fitting. After the fitting, the model is first used on the test set of the first repetition, calculating the score for the test set.

It is important to state that the downstream machine learning model is only trained once, during the first repetition of the experiment. For the second and third repetitions, the program reuses the aforementioned downstream model on the test sets of the remaining two repetitions. As mentioned before, these test datasets consist of the same data points, but the exact data points that are perturbed change in every repetition. After each repetition, the results are documented in the respective CSV file in the user-defined location. The program creates therefore separate CSV files for the results of each repetition, as well as a summary in which the scores are averaged for all repetitions. The procedure is also conducted for the recorded time it takes to train the imputation model for each repetition.

6.3. Imputation Experiments on Data Subset

In addition to the regular imputation experiments, tests are conducted on subsets of the datasets. Since the training of the imputation model and the downstream machine learning model can consume a lot of resources and time, the question arises, how the training time can be reduced. The exact training times for the imputation models are defined in section 7.5. A possibility to achieve this goal is to reduce the size of the dataset. The idea behind this approach is to test, whether it is possible to conduct the same experiments on a subset of the dataset and achieve similar results as in the imputation experiments on the complete datasets.

Therefore, the author added an additional split after the dataset is pulled from OpenML. The program randomly chooses 10% of the data points of the dataset and

drops the remaining 90%. After this, the experiment is conducted as described in the previous section. To ensure that the experiments on the subset of the dataset are deterministic, a fixed random seed is added.

6.4. Automated Recommendation Based on Dataset Properties

The central topic of this thesis is to assess and predict the optimal imputation method for a given missing data scenario, keeping future usage as training data for a machine learning model in mind. Since it has become clear from the literature research in chapter 3 and from the results of the imputation experiments in section 7.1, that there is no absolute best imputation method, that outperforms the other imputation methods clearly in every scenario, the necessity to select the right imputation method for the given dataset and situation remains.

The imputation experiments conducted in this thesis cover a variety of different scenarios with a total of 4824 experiments, depicting 804 different scenarios over 67 datasets. The results of these experiments provide a useful database, that can be facilitated for further approaches. All scenarios are in summary unique and each of these scenarios has one explicit imputation method, that performed the best for this particular situation.

Based on this information, it is possible to detect similarities between a user-given dataset with missing data and the different scenarios covered in the imputation experiments in this thesis. With this realization, it is possible to train a machine learning model, which should automatically detect the ideal imputation method for a given dataset. To train the model for this classification task, the author considered several features, which make it possible for the model to sufficiently distinguish between the different scenarios and provide a credible prediction. The features have to be available for a practitioner, who only has the dataset at hand. Therefore the following features are used for the training:

- Number of Data Points
- Number of Categorical Features
- Number of Numerical Features
- Missingness Type
- Missing Fraction

And in addition to that as label:

- Imputation Method

It is important to note at this point, that this kind of experiment is only conducted for the two classification tasks and not for the regression task. The reason for this is that there is barely any noticeable significant improvement from the other imputation methods relative to the average best imputation method in that segment. Further details can be found in the subsection 7.1.3.

The improvement in F1 score points between the best imputation method per data constellation and the average best imputation method overall often lies beneath 0.01 or 0.03 F1 score points. Therefore the datasets for this experiment have been altered to create in total three versions. For the first version, the original dataset with only the best imputation method per data constellation as the label is used. The second version (0.01 F1 Adjustment) features a slightly altered dataset. Every time, the best imputation method outperforms the average best imputation method by less than 0.01 F1 score points, the best imputation method is replaced by the average best method as label for that particular data constellation. The third version features a similar change, only for 0.03 F1 score points (0.01 F1 Adjustment).

For this model, the author facilitates the random forest decision tree as a machine learning model approach. The data used for training is preprocessed by one-hot encoding the missingness pattern (Missingess Type) and converting all features into numpy arrays. Since there are only few data points available, 372 for binary classification and 204 for multiclass classification, the author facilitates nested cross-validation. For the inner split, a five-fold split is used, while for the outer split, a ten-fold split is utilized. Since for some versions, the dataset becomes increasingly unbalanced, due to the majority of labels being the average best method, the hyperparameter *class_weight* is set to *'balanced'*. This mode automatically adjusts the class weights inversely proportional to class frequencies in the dataset ("sklearn RandomForestClassifier", 2023). To ensure reproducibility the random seed for the model remains static for all experiments.

For further tuning of the hyperparameters *n_estimators* and *max_depth* a parameter grid is implemented in combination with scikit-learn's GridSearchCV module.

After conducting the nested cross-validation, the suggested predictions on the test data for the best model from the outer split are taken and their F1 score from the imputation experiments is compared to the F1 score of the average best method from the imputation experiments on that particular data constellation. Then, the average difference in F1 score between the suggestion from the model and the average best method is calculated. The result shows whether the models' suggestions outperform

on average the simpler approach, by which the practitioner simply applies the determined average best imputation for all scenarios.

7. Results

In this chapter, the author summarizes the results of the experiments described in the previous chapter. Since the single results from the different scenarios are often difficult to compare directly with each other, especially across datasets, a ranking system is applied. First, the program determines the average F1 Score or RMSE on the test set, as explained in subsubsection 4.2.2, across all three repetitions of one experiment. This leads to in total six different results for each data constellation scenario, one for each imputation method. During the subsequent analysis, these results are ranked from one to six, one being the best result, and six the worst. Aside from this basic approach, there are additional aspects to consider. There are a few instances, especially for small datasets with a low number of missing valuesw, where the scores are identical. In these situations, the imputation methods with lower computational costs are given priority over the more time-consuming approaches. Based on the results described in section 7.5, the following order is applied in these cases:

i) Mean/Mode

ii) K-NN

iii) Random Forest

iv) VAE

v) GAIN

vi) Discriminative Deep Learning

Aside from those instances, there are situations in which the training of the imputation model fails, which automatically results in being assigned the worst rank. Most notably for the generative deep learning approach GAIN. In total 57 experiments, 7% of all experiments for GAIN on complete datasets, the training for the imputation model failed. Jäger et al. (2021) described in their scientific paper similar problems with this method and attributed these issues to the GAIN discriminator loss getting a NAN value at some point of the training, leading to failures on further calculations and ultimately a failing training process. They further state, that this depends on the discriminators' learning rate and the given dataset. Since GAN-based models are widely known to be hard to train, Jäger et al. (2021) suggested a predefined

hyperparameter grid, as described in subsubsection 4.2.2, since other improvements, suggested by other researchers to make the training more robust, were out of scope for their particular research, as well as for this thesis. Aside from the GAIN imputation method, the training failed for K-NN (5 times), random forest (12 times), and discriminative deep learning (19 times) as well. While the issues for K-NN and random forest can be contributed to a particular dataset, which apparently caused excessive memory usage resulting in the operating system having to kill a worker process, the issues surrounding the discriminative deep learning approach appear less clear. During the tests to recreate the results from Jäger et al. (2021), as well as during the imputation experiment for this thesis, problems appeared, which are not mentioned by Jäger et al. (2021) in their scientific paper. Due to issues in the early stages of the thesis with Jenga and the data imputation framework, these problems can be attributed to compatibility issues for the used machine learning libraries tensorflow and autokeras with several other required libraries and drivers, for instance, the drivers for the GPU of the used hardware for these experiments. Since Jäger et al. (2021) and Schelter et al. (2021) were facilitating older versions of several libraries, it is not feasible or within the scope of this thesis to downgrade and test all required libraries to use those particular versions, especially since the aforementioned authors did not disclose those particular versions of the facilitated libraries for their respective frameworks. Given the fact, that only 19 out of 804 imputation experiments are affected by this issue, the consequences can be considered negligible.

As described before, the experiment scenarios are comprised of three different missingness patterns and four different missing fractions for each dataset and imputation method. An example of such a data constellation can look like the following:

Dataset ID: 737; Missingness Pattern: MAR; Missing Fraction: 0.3; Imputation Method: K-NN

This will also be referred to as an experiment, scenario, experiment setting, data constellation, or situation in the following sections of this chapter.

The following sections depict the results of the imputation experiments on the complete datasets and the subsets of those datasets, as well as comparisons between them and the baseline results for the experiments on uncorrupted data. Therefore the author investigates which method performs the best most frequently, which method performs the best on average, and how big the differences are between those. Most of the previously described literature researched, whether it is possible to determine if there is a best imputation method and if so, which one it would be. To possibly answer this question, a closer look will be undertaken to see how far off the average best imputation method is, from the actual best method for each data constellation in this experimental setup.

The following sections feature several figures to visualize the results of the experiments. These figures can be categorized into three different types. The first type depicts the results of all experiments of the respective section, excluding experiment results for the average best imputation method. The results are displayed in a bar chart. The stacks represent the number of experiment results, which achieved a certain improvement or deterioration relative to the average best method. The x-axis shows the ranges, defined for that. Classification results are always presented as F1 score points. The results for the regression experiments are shown in percentage (eg. 0.09 equals 9%). The reason for this is, that the RMSE can vary greatly between different datasets, depending on the values in the to-be-imputed column. The second type depicts only the results for the best downstream machine learning model performance for each experiment data constellation in the respective section and their improvement relative to the average best method on each particular data constellation. For further evaluation, the stacks for both bar charts can be split to represent the distribution for either the imputation methods, missingness patterns, or missing fractions.

The third kind of figure is the heatmap. The heatmaps show the experiment results for each imputation method for each dataset in their specific section. One heatmap shows therefore one particular missingness pattern and one specific missing pattern each. It is also important to note, that not the absolute scores are depicted, but rather the scores relative to a different result, for instance to the baseline model or the average best imputation method. This information depends on the particular chapter and the attached description of the respective figure.

Some of the figures presented in this chapter feature a zoomed-in graph, which means not the whole graph is visible. This is usually done to outline a certain argument or to make the results easier to assess for the reader. Every chart, that is created from the experiments, is attached in its completeness in the appendix of this thesis.

7.1. Imputation Experiments

This section shows the results of the imputation experiment separated for the different downstream machine learning tasks.

7.1.1. Binary Classification

For the binary classification, 31 datasets are facilitated, consisting of 3107 to 96320 data points and up to 24 features with the majority of those being numerical. This leads to 372 experiment scenarios, which are executed for all described imputation

methods, resulting in 2232 experiments in total.

As portrayed in the following Table 7.1, random forest performs on average the best for binary classification with an average rank of exactly 3.0. In correlation with this outcome, random forest is also the imputation method, which delivers the best result most frequently, in total on 89 occasions. While K-NN is not far off in terms of the average rank, it loses out for the best performance per scenario not only to random forest, but also to the mean/mode approach. Discriminate deep learning is rather close in terms of the average rank while the two generative deep learning approaches lose out to the simpler imputation methods in both categories.

Imputation Method	Average Rank	Rank 1 Occurrences
Mean/Mode	3.4086	69
K-NN	3.1505	61
Random Forest	3.0	89
Discriminative DL	3.3978	52
VAE	3.7016	54
GAIN	4.3602	47

Table 7.1.: Imputation Results Overview for Binary Classification

The difference in the absolute F1 score from the best imputation method per data constellation and the average best imputation method (random forest) for binary classification is on average 0.0108 F1 score points. This amounts effectively to an average improvement of 1.9%, if the best imputation method is chosen instead of the average best method.

Further analysis shows the distribution of the impact on the predictive performance of the downstream tasks relative to the random forest approach. The Figure 7.1 portrays the differences in F1 score points of all binary classification experiments for all imputation methods, excluding random forest.

It illustrates that the vast majority of the results are rather similar to the average best methods performance. More than 75% of all outcomes are within a 0.01 F1 points improvement or a 0.01 F1 points deterioration of the respective result of the average best method for their particular data constellation. It is important to note that there are some significant outlier scenarios with either more than 0.09 F1 points improvement or deterioration.

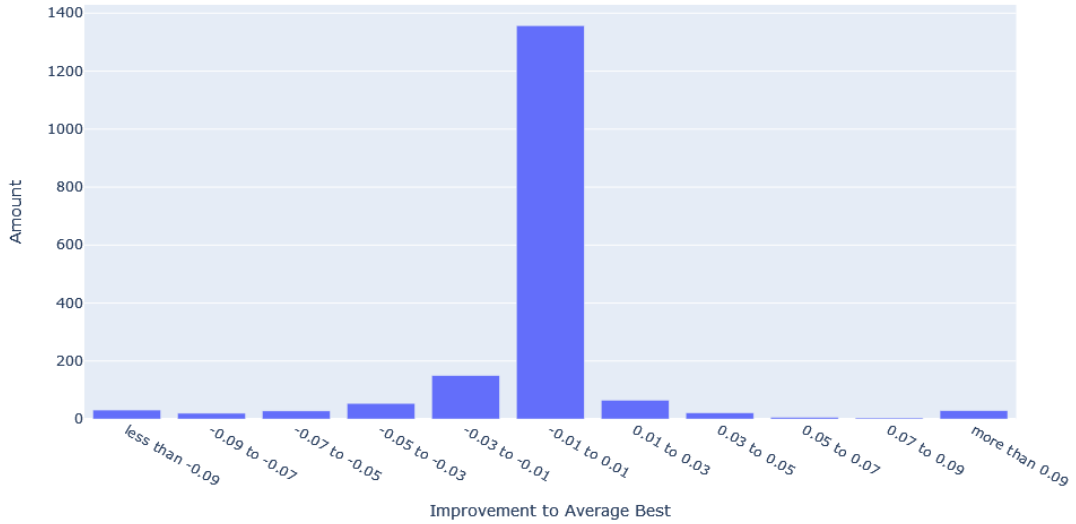

Figure 7.1.: Improvement for Imputation Experiments Relative to the Average Best Imputation Method by F1 Score Points

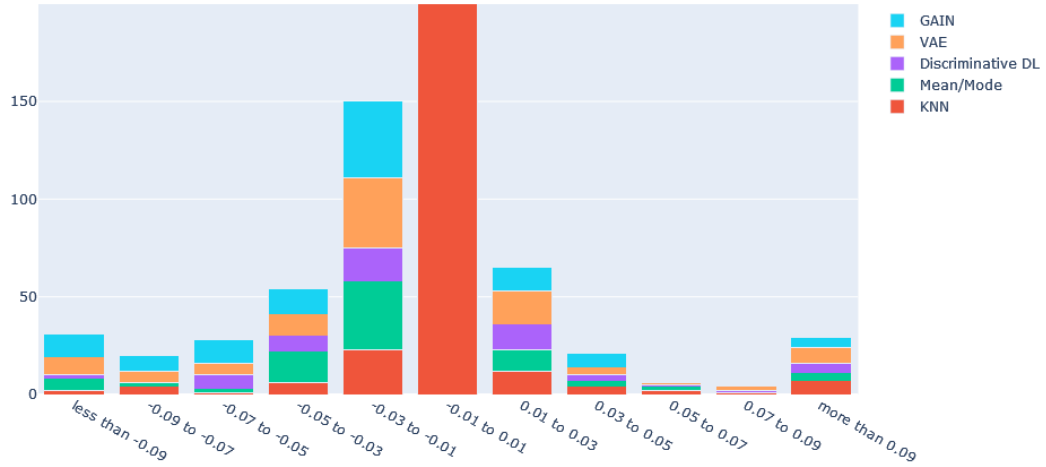

Figure 7.2.: Improvement for Imputation Experiments Relative to the Average Best Imputation Method by F1 Score Points by Imputation Methods (Zoomed)

Apart from the majority of severe deteriorations originating from the generative deep learning approaches, there seems to be no clear trend among the imputation

methods themselves (Figure 7.2).

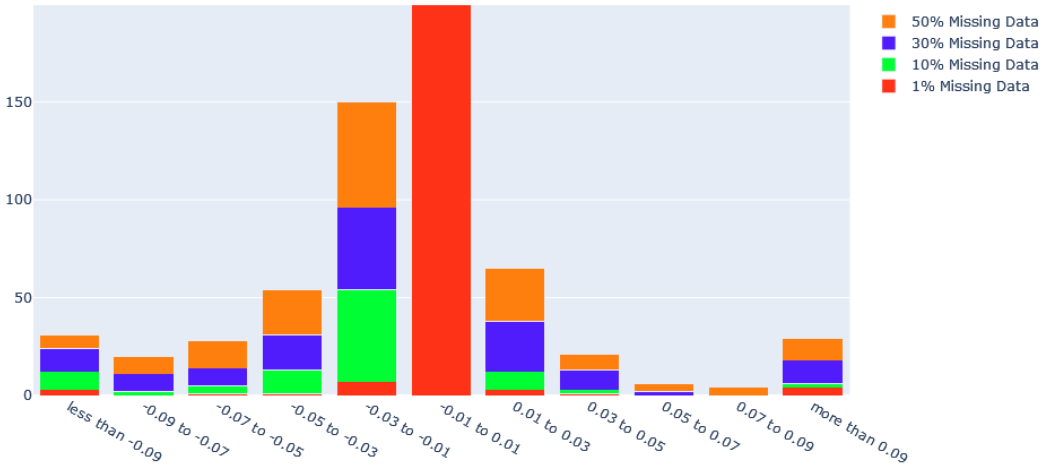

Figure 7.3.: Improvement for Imputation Experiments Relative to the Average Best Imputation Method by F1 Score Points by Missing Fraction (Zoomed)

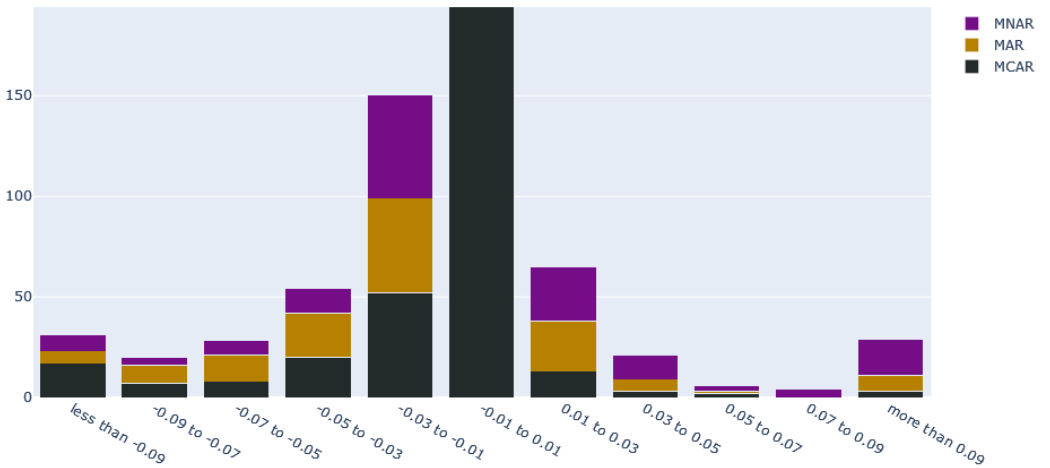

Figure 7.4.: Improvement for Imputation Experiments Relative to the Average Best Imputation Method by F1 Score Points by Missingness Pattern (Zoomed)

It appears that the majority of significant improvements can be traced back to experiments with a higher fraction of missing data (Figure 7.3) as well as an MNAR missingness pattern. In contrast, the more significant deteriorations seem connected to MCAR missingness situations, as depicted in Figure 7.4.

To get some more detailed information regarding the matter of how significant the difference between using the best imputation method per data constellation or just simply using random forest is, the following Figure 7.5 illustrates all imputation results, that are ranked the best for their respective data constellation, including random forest.

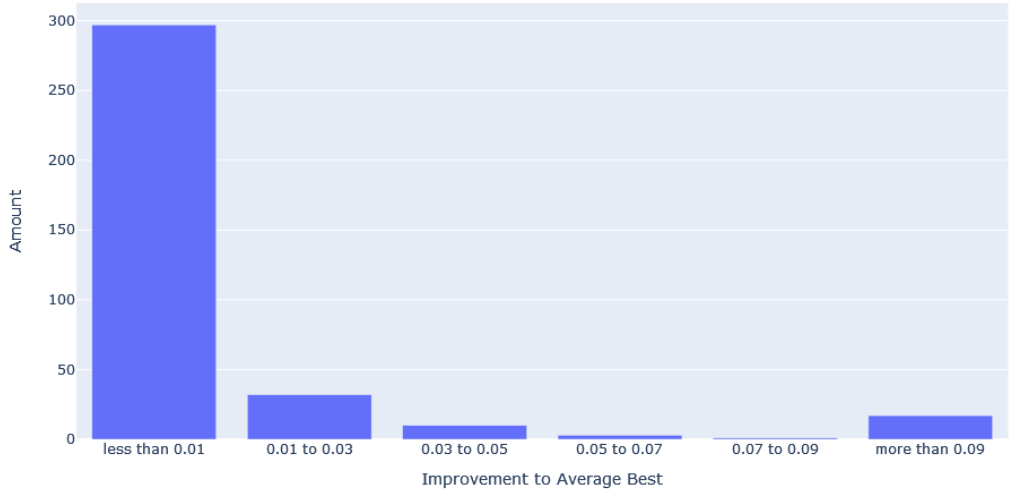

Figure 7.5.: Improvement for Best Imputation Method per Experiment Scenario Relative to the Average Best Imputation Method by F1 Score Points

It can be seen in the Figure 7.5, that there are several cases, in which the best suited method improves the predictive performance of the downstream machine learning task quite significantly by at least 0.09 F1 score points. These cases comprise 4.6% of all the experiments in this section. Further, it is visible that in 8.3% (31 cases) of experiment scenarios the result of the best imputation method exceeds the random forest approach by at least 0.03 F1 score points.

While the is no clear pattern for a particular imputation method visible, it appears that if a higher fraction of the data is missing (Figure 7.5), as well as if the missingness pattern isn't MCAR (Figure 7.7), that there is also a higher potential for improvement.

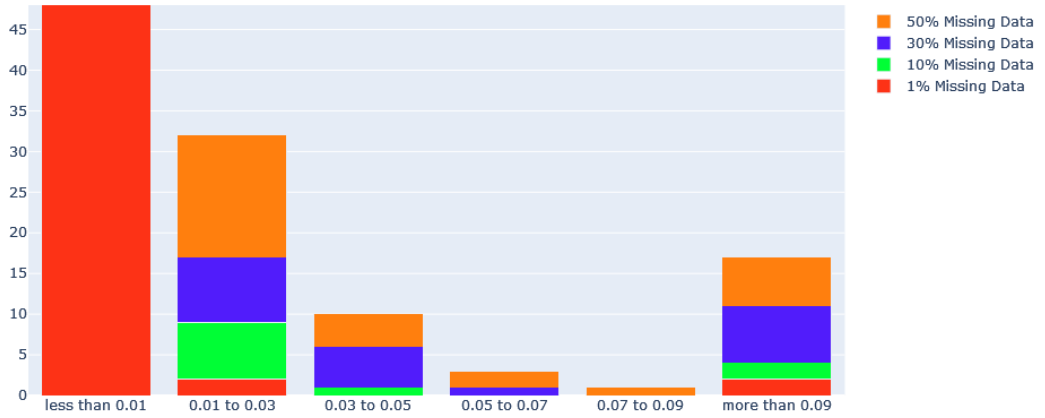

Figure 7.6.: Improvement for Best Imputation Method per Experiment Scenario Relative to the Average Best Imputation Method by F1 Score Points by Missing Fraction

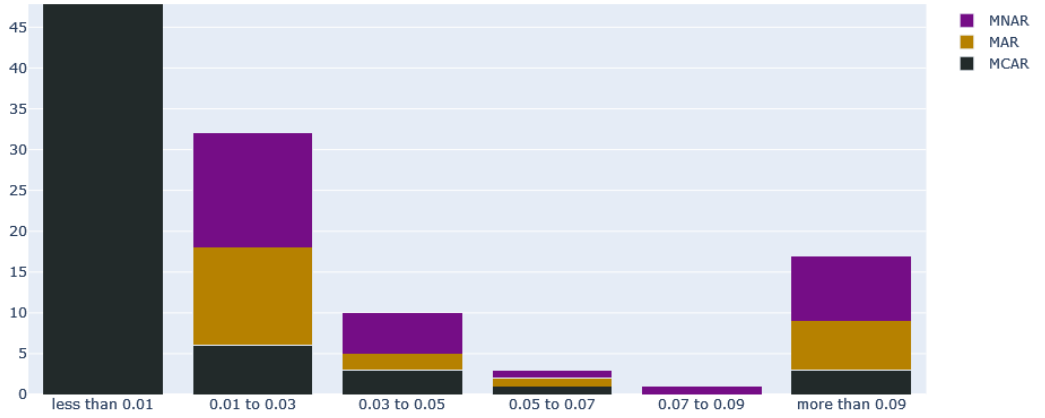

Figure 7.7.: Improvement for Best Imputation Method per Experiment Scenario Relative to the Average Best Imputation Method by F1 Score Points by Missingness Pattern

7.1.2. Multiclass Classification

For the multiclass classification 17 datasets are used, comprised of 3200 to 58000 data points and up to 25 features with the majority of those being numerical. This leads to 204 experiment scenarios, which are conducted for all six imputation methods, resulting in 1224 experiments in total.

The following Table 7.2 shows the results for the multiclass classification experiments. As for the binary classification, random forest achieves on average the best rank with 3.27. In contrast to the binary classification, the average ranks of all imputation methods are relatively close, ranging from 3.2 to 3.8. This is supported by the distribution of rank one occurrences. While VAE scores the most with 41 number one rankings, aside from discriminative deep learning, all other methods score around 35 number one rankings.

Imputation Method	Average Rank	Rank 1 Occurrences
Mean/Mode	3.3676	35
K-NN	3.3235	36
Random Forest	3.2745	34
Discriminative DL	3.7549	23
VAE	3.4803	41
GAIN	3.8039	35

Table 7.2.: Imputation Results Overview for Multiclass Classification

The difference in the absolute F1 score from the best imputation method per data constellation and the average best imputation method (random forest) for multiclass classification is on average 0.0153 F1 score points. This amounts effectively to an average improvement of 4.19%, if the best imputation method is chosen instead of the average best method.

Further insights can be gained by taking a closer look at the distribution of the different results from the analysis, relative to the average best imputation method (random forest). The following Figure 7.8 depicts this distribution by F1 score points, excluding the random forest experiments, which looks very similar to a classic Gaussian distribution. 54% of the experiments of the other five imputation methods achieve rather similar results to the random forest approach, their improvements or deteriorations are within 0.01 F1 score points, relative to the random forest outcomes. Aside from those, there are few experiments that yield a result in which one of the other imputation methods is capable of significantly outscoring the random forest approach.

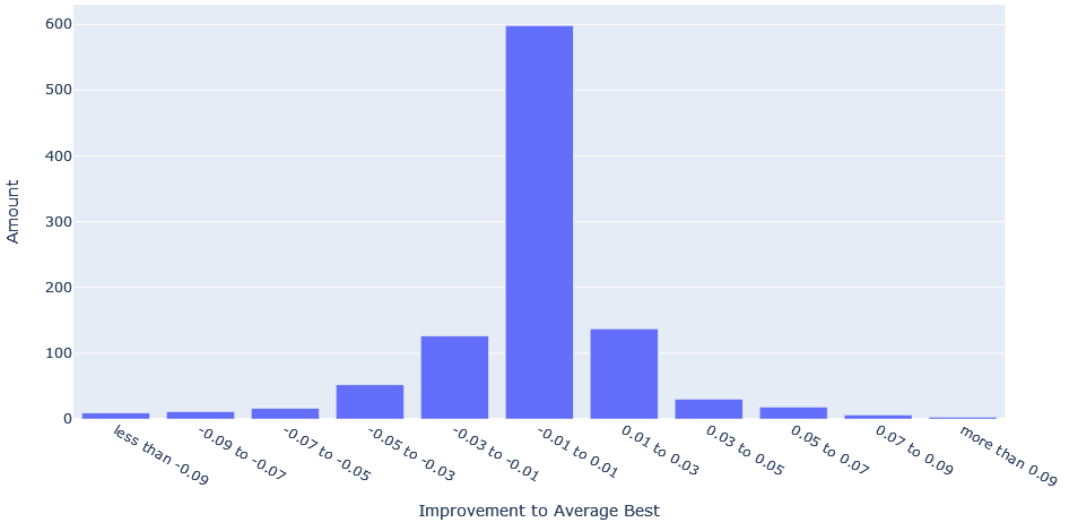

Figure 7.8.: Improvement for Imputation Experiments Relative to the Average Best Imputation Method by F1 Score Points

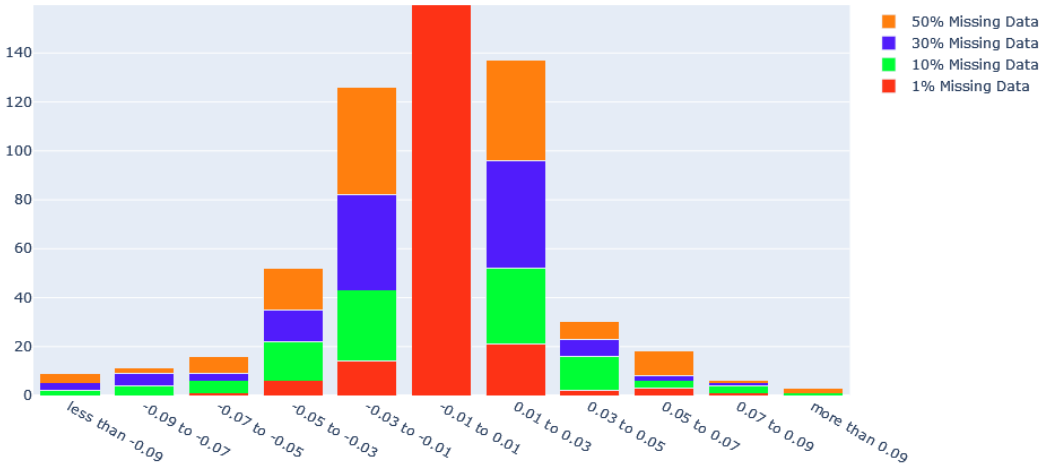

Figure 7.9.: Improvement for Imputation Experiments Relative to the Average Best Imputation Method by F1 Score Points by Missing Fraction (Zoomed)

The Figure 7.9 is a cut-out from the same Figure 7.8 as seen above, where the bars of the bar chart are divided into the missing fractions of the represented

experiments. Zooming in on the lower regions of the bar chart allows a better view of the proportions depicted in the figure. It can be seen, that for significant improvements or deteriorations, a larger amount of missing data is often required. Imputing only one percent of the data of a dataset, especially for large datasets, does not seem to have a significant enough impact on the downstream predictive performance, to massively improve or worse their test scores.

A closer look at the distribution for the imputation methods or the applied missingness patterns does not reveal any further trends or insights.

To get more detailed information regarding the significance of the difference between using the best imputation method per data constellation or just simply using random forest, the following Figure 7.10 illustrates all imputation results, that are ranked the best for their respective data constellation, including random forest. It appears to make more of a difference whether random forest or the respective best imputation method is chosen, although the majority of improvements remain below 0.03 F1 score points. 14.7% of experiment scenarios lead to a situation in which the respective best imputation method outscores the average best approach for their respective experiment setting by more than 0.03 F1 score points.

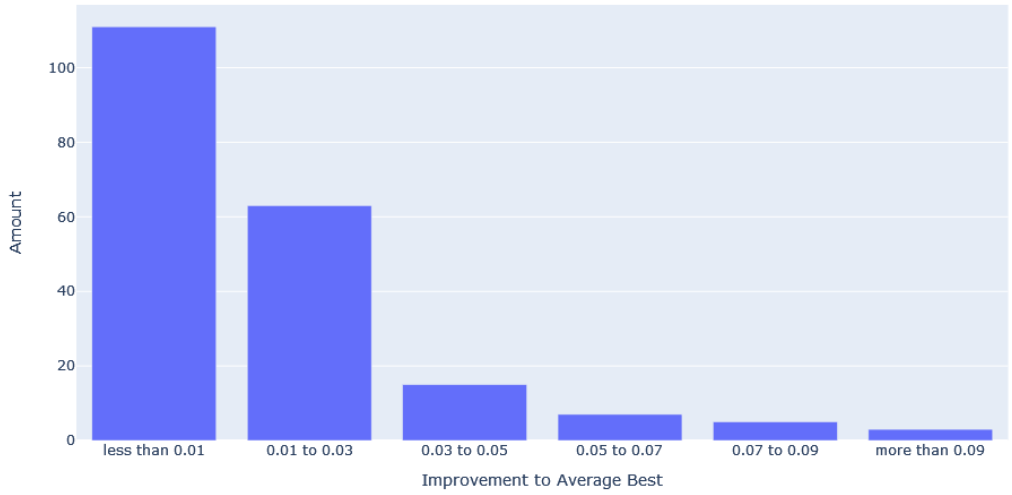

Figure 7.10.: Improvement for Best Imputation Method per Experiment Scenario Relative to the Average Best Imputation Method by F1 Score Points

Looking further it appears that these significant improvements are mainly achieved by the generative deep learning imputation approaches, VAE and GAIN. Simpler imputation methods like mean/mode and K-NN mostly remain below 0.05 F1 score points with their improvements. The following Figure 7.11 depicts this outcome and

shows the further distribution of the level of improvement relative to the random forest approach by imputation method.

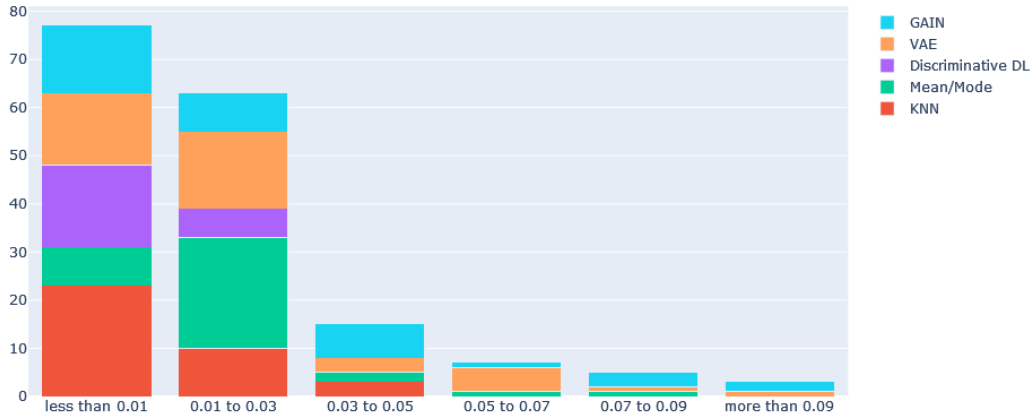

Figure 7.11.: Improvement for Best Imputation Method per Experiment Scenario Relative to the Average Best Imputation Method by F1 Score Points by Imputation Method

Among the missingness patterns and the missing fractions, there is no clear trend visible, the results are rather balanced between the different settings.

7.1.3. Regression

For the regression tasks, 19 datasets are used, consisting of 4477 to 95911 data points and up to 22 features with the majority also being numerical. This results in 228 experiment scenarios, which are conducted for all six imputation methods, leading to 1368 experiments in total.

The Table 7.3 below shows the results for the regression experiments. As for the two classification task types, random forest achieves on average the best rank with an average ranking of 3.10. Aside from the two generative imputation methods, the remaining four imputation approaches are again rather close in terms of average assigned rank. Despite only attaining 24 number one rankings, K-NN still reaches the same average rank as the mean/mode approach, despite the latter scoring 50 number one rankings, the most for regression tasks. This indicates rather high fluctuations between the predictive performance of the downstream models trained

on mean/mode imputed data.

Imputation Method	Average Rank	Rank 1 Occurrences
Mean/Mode	3.3026	50
K-NN	3.3026	24
Random Forest	3.1052	48
Discriminative DL	3.1228	37
VAE	3.7982	31
GAIN	4.3684	38

Table 7.3.: Imputation Results Overview for Regression

The difference in the absolute RMSE score from the best imputation method per data constellation and the average best imputation method for regression (random forest) is on average 0.8315. This amounts effectively to an average improvement of 0.24%, if the best imputation method is chosen instead of the average best method.

For further insights, a closer look at the distribution of the different results from the analysis, relative to the average best imputation method (random forest) can be undertaken. The following Figure 7.12 depicts this distribution, excluding the random forest experiments as the average best method.

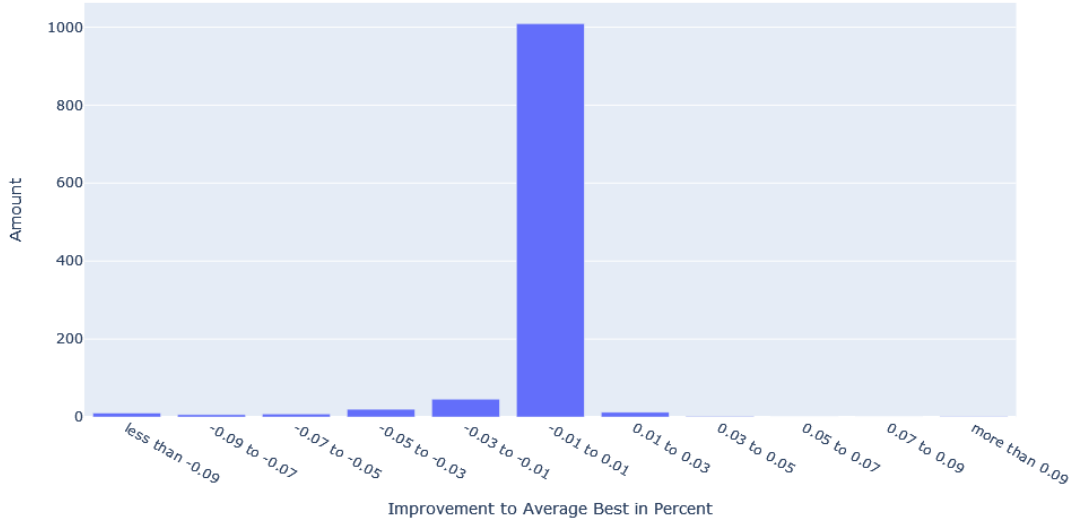

Figure 7.12.: Improvement for Imputation Experiments Relative to the Average Best Imputation Method in Percent

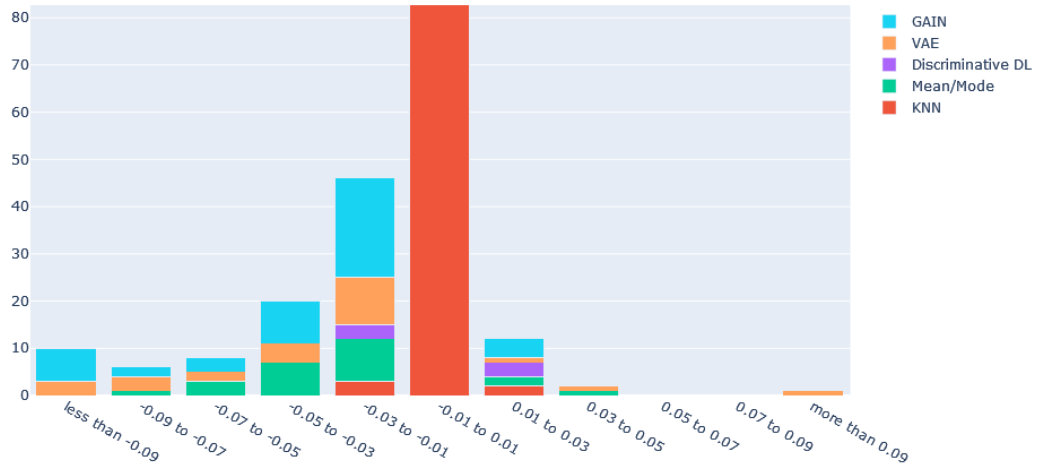

Figure 7.13.: Improvement for Imputation Experiments Relative to the Average Best Imputation Method in Percent by Imputation Method (Zoomed)

It shows that the differences between the six imputation methods are rather small, given that more than 90% of the experiments with imputation methods other than random forest are achieving results within 1% improvement and 1% deterioration relative to their respective random forest result. In total only three experiment results exceed the improvement threshold relative to the average best method by more than 3% and only one of those outcomes reaches more than 9% improvement.

It appears that there is a higher potential for deteriorating results relative to the average best imputation method for higher missing fractions and the generative deep learning imputation methods, as visible in the following Figure 7.13, which shows a zoomed-in cut-out of the previously shown Figure 7.12, decoded by imputation approach per bar chart stack.

Like in the previous sections, to get more detailed information regarding the importance of the difference between using the best imputation method per data constellation or using random forest, the following Figure 7.14 illustrates all imputation results, that are ranked the best for their respective data constellation, including random forest. This bar chart confirms the results described before in this section, meaning there are barely any notable improvements in downstream prediction performance for the other imputation methods in comparison to the random forest approach.

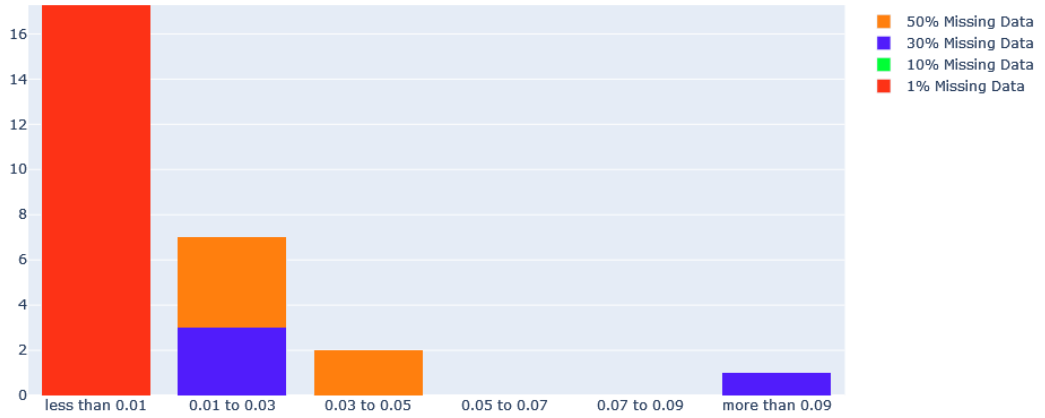

Figure 7.15.: Improvement for Best Imputation Method per Experiment Scenario Relative to the Average Best Imputation Method in Percent by Missing Fraction (Zoomed)

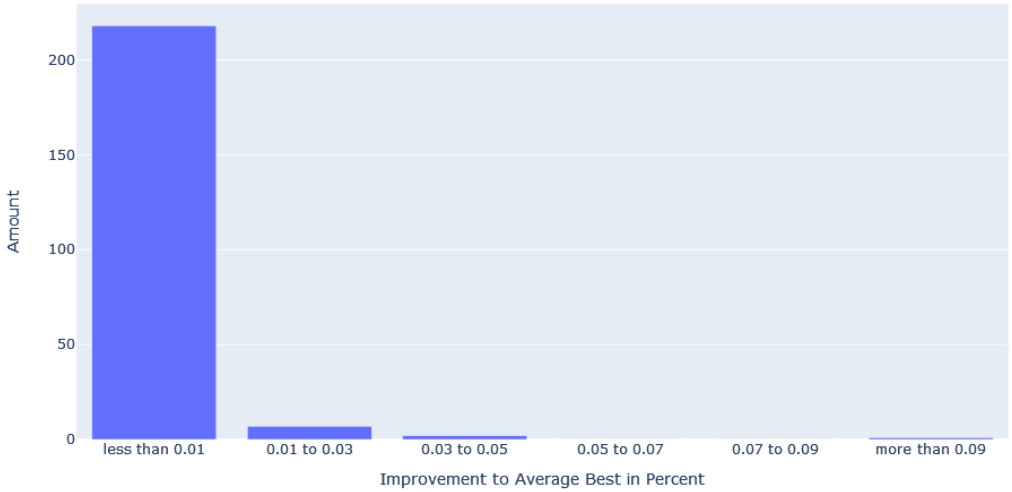

Figure 7.14.: Improvement for Best Imputation Method per Experiment Scenario Relative to the Average Best Imputation Method in Percent

It is worth mentioning, that only experiment scenarios with 30% or 50% missing data are achieving improvements above 1% (Figure 7.15). Aside from that, there are no notable trends among the imputation methods or missingness patterns, mainly due to the fact that there are very few significant improvements at all.

7.2. Imputation Experiments on Data Subset

This section shows the results of the imputation experiments on the data subsets, separated for the different downstream machine learning tasks. The experiment settings regarding the datasets remain the same as for the full datasets, with the exception, that only 10% of the data points for each dataset are used.

7.2.1. Binary Classification

As portrayed in the following Table 7.4, mean/mode imputation performs on average the best for binary classification on a data subset with an average rank of 2.86. In accordance with this outcome, mean/mode is also the imputation method, which achieves the best result most frequently, in total on 101 occasions. This is the only area in which the average best method achieves an average rank better than 3.0 and gets ranked best more than a hundred times. While the simpler imputation methods, K-NN and random forest, are still achieving competitive results for their average ranks and number one ranking occurrences, the more complex imputation approaches, discriminative deep learning, VAE, and GAIN, seem to be having trouble dealing with the smaller subsets. Those methods lose out by almost a whole ranking position for the average rank and have significantly fewer number one rankings than the other three imputation approaches.

Imputation Method	Average Rank	Rank 1 Occurrences
Mean/Mode	2.8655	101
K-NN	3.1639	66
Random Forest	3.2956	68
Discriminative DL	3.9489	49
VAE	3.7096	46
GAIN	4.0161	42

Table 7.4.: Imputation Results Overview for Binary Classification (Subset)

The difference in the absolute F1 score from the best imputation method per data constellation and the average best imputation method (mean/mode) for binary classification on data subsets is on average 0.0176 F1 score points. This amounts

effectively to an average improvement of 2,87%, if the best imputation method is chosen instead of the average best method.

Further analysis shows the distribution of the impact on the predictive performance of the downstream tasks relative to the mean/mode approach. The following **??** portrays the differences in F1 score points of all binary classification experiments for the data subsets for all imputation methods, excluding mean/mode. It reveals an almost typical Gaussian distribution, but with more extreme cases where the improvement or deterioration exceeds 0.09 F1 score points. Around 63% of all outcomes are within a 0.01 F1 points improvement or a 0.01 F1 points deterioration of the respective result of the average best method for their particular data constellation. Similar to the complete dataset experiments, there are some significant outlier scenarios with either more than 0.09 F1 points improvement or deterioration. Generally, there appear 123 scenarios with more than 0.03 F1 points improvement and 187 scenarios with more than 0.03 F1 points deterioration.

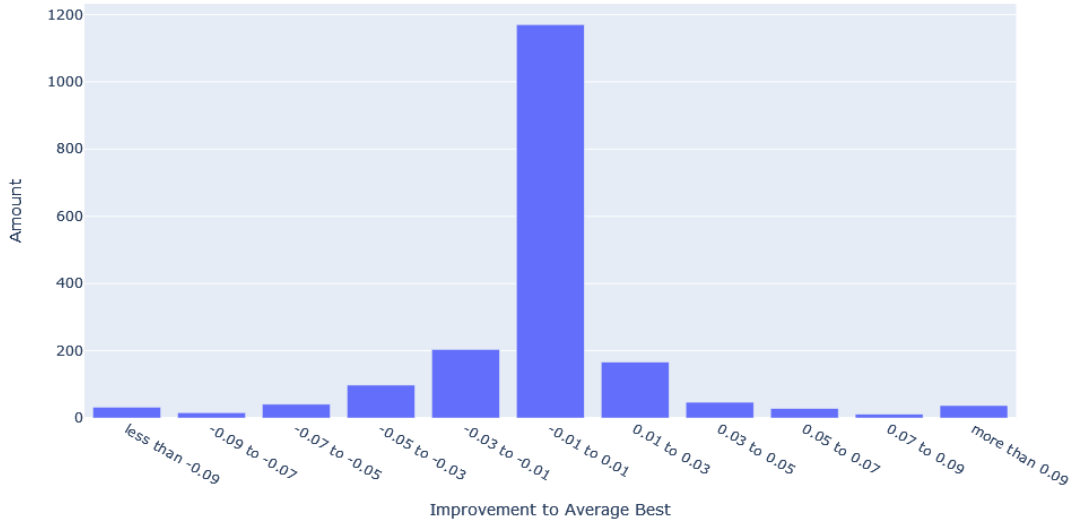

Figure 7.16.: Improvement for Imputation Experiments Relative to the Average Best Imputation Method by F1 Score Points

The majority of the more significant performance differences in both directions can be attributed to higher fractions of missing data, although there is a notable portion of 14 scenarios with only one percent missing data, that achieve more than 0.09 F1 score points improvement relative to their average best method for those scenarios (Figure 7.17).

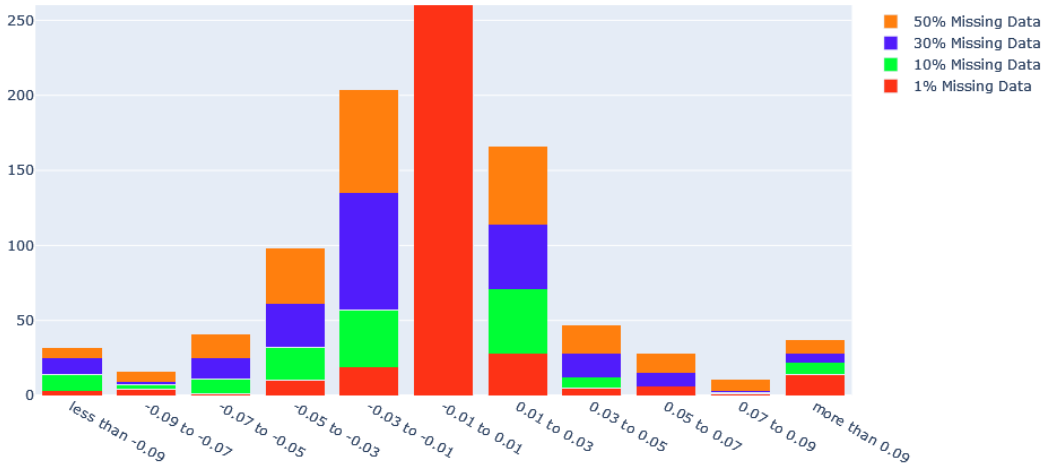

Figure 7.17.: Improvement for Imputation Experiments Relative to the Average Best Imputation Method by F1 Score Points by Missing Fraction (Zoomed)

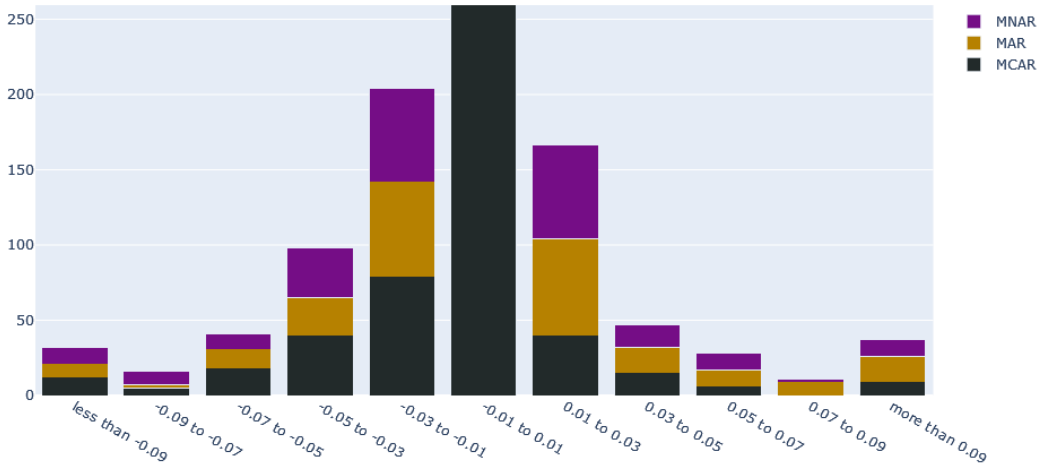

Figure 7.18.: Improvement for Imputation Experiments Relative to the Average Best Imputation Method by F1 Score Points by Missingness Pattern (Zoomed)

Another observation regarding these experiments reveals that there is a trend toward the MAR missingness pattern in regard to improvements, as seen in the Figure 7.18.

To get more detailed information regarding the matter of how significant the difference is, between using the best imputation method per data constellation or just simply using random forest, the following Figure 7.19 illustrates all imputation results, that are ranked the best for their respective data constellation, including mean/mode. It can be seen in the following Figure 7.19, that there are several cases, in which the best suited method improves the predictive performance of the downstream machine learning task quite significantly by at least 0.09 F1 score points. These cases comprise 4.8% of all the experiments in this section. Furthermore, it is visible that in 16.6% (62 cases) of experiment scenarios the result of the best imputation method exceeds the mean/mode approach by at least 3%. This is double the amount than for the imputation experiments on the complete datasets and shows that there is certainly potential for improvement.

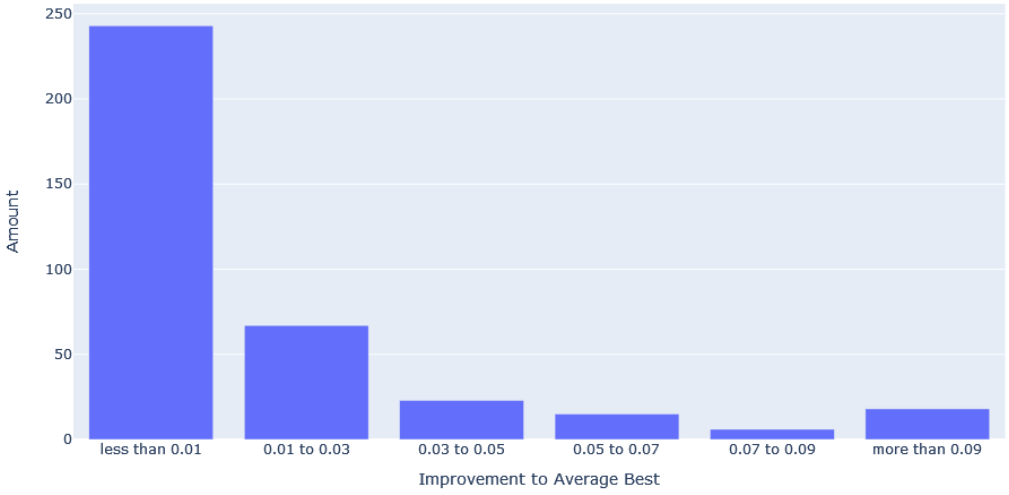

Figure 7.19.: Improvement for Best Imputation Method per Experiment Scenario Relative to the Average Best Imputation Method by F1 Score Points

For this part of the analysis, it sticks out, that the majority of the improvement scenarios can be attributed to the simpler imputation methods, K-NN and random forest, as seen in the Figure 7.20 below.

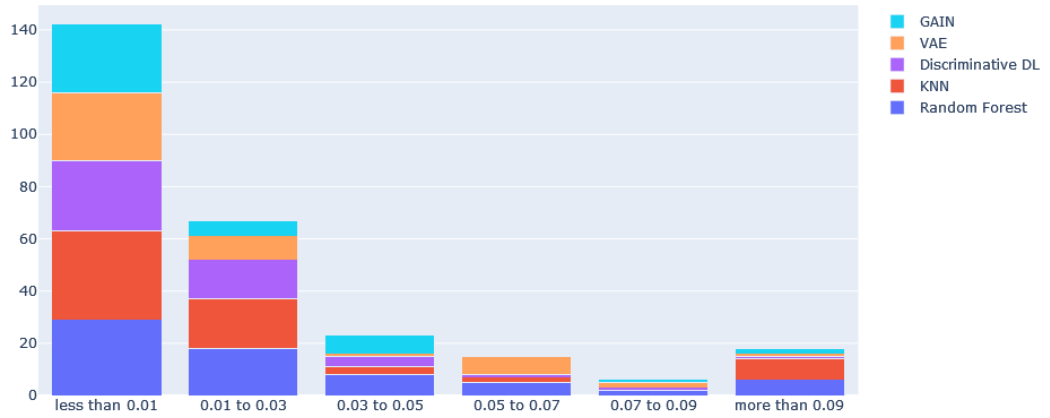

Figure 7.20.: Improvement for Best Imputation Method per Experiment Scenario Relative to the Average Best Imputation Method by F1 Score Points by Imputation Method

7.2.2. Multiclass Classification

The following Table 7.5 shows the results for the multiclass classification experiments on the data subsets. Unlike the observations from the experiments on the full datasets, these results depict clearer differences between the downstream task performances for the different imputation methods. K-NN achieves on average the best rank with 3.09. The margin to the next best methods, mean/mode and random forest, is already considerable since they only achieve an average rank of around 3.3. Again, the discriminative and generative deep learning approaches lose out in both categories to their simpler counterparts.

Imputation Method	Average Rank	Rank 1 Occurrences
Mean/Mode	3.2941	38
K-NN	3.0931	50
Random Forest	3.3137	33
Discriminative DL	3.9264	24
VAE	3.7549	32
GAIN	3.6176	27

Table 7.5.: Imputation Results Overview for Multiclass Classification (Subset)

The difference in the absolute F1 score from the best imputation method per data constellation and the average best imputation method for multiclass classification on the data subsets (K-NN) is on average 0.0252 F1 score points. This amounts effectively to an average improvement of 6.08%, if the best imputation method is chosen instead of the average best method.

Further insights can be gained by taking a closer look at the distribution of the different results from the analysis, relative to the average best imputation method (K-NN). The following Figure 7.21 depicts this distribution, excluding the K-NN experiments. It shows that there are several scenarios in which a significant improvement is possible, but even more scenarios in which a severe deterioration of the predictive performance of the downstream machine learning model is apparent.

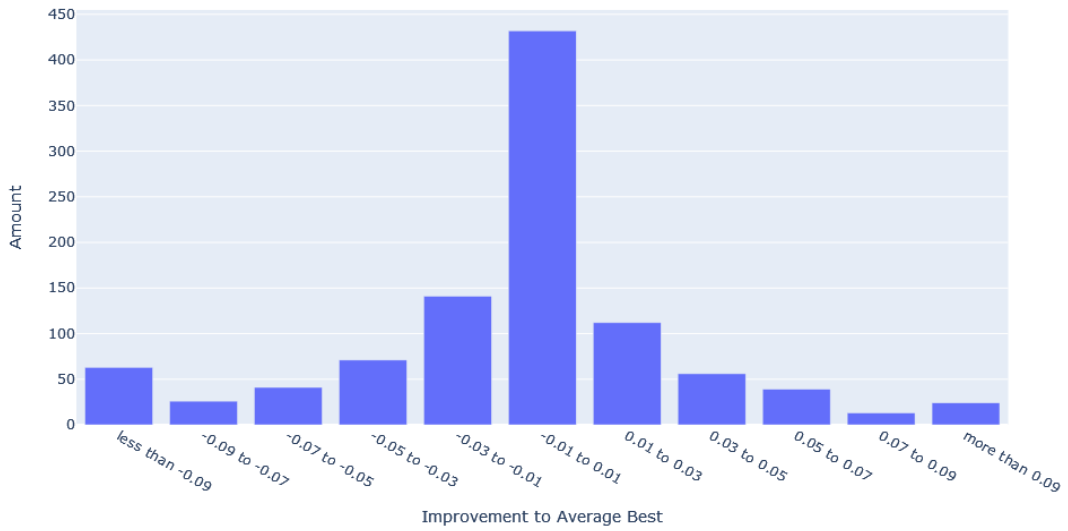

Figure 7.21.: Improvement for Imputation Experiments Relative to the Average Best Imputation Method by F1 Score Points

While there are no clear patterns for the distribution of the imputation methods or missingness patterns in that regard, it is surprising to see in the following, zoomed in bar chart in Figure 7.22, that the majority of severe improvements or deteriorations originate from experiments scenarios with low fractions of missing data.

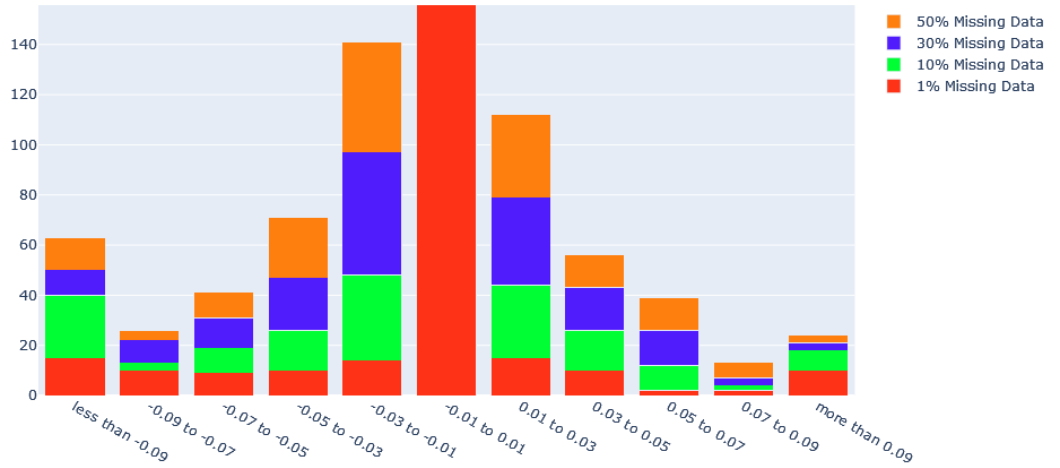

Figure 7.22.: Improvement for Imputation Experiments Relative to the Average Best Imputation Method by F1 Score Points by Missing Fraction (Zoomed)

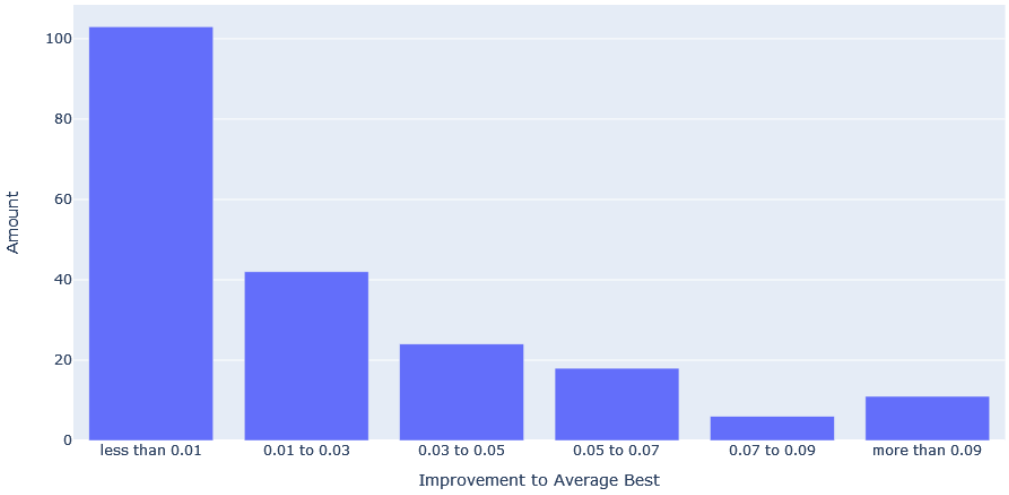

Figure 7.23.: Improvement for Best Imputation Method per Experiment Scenario Relative to the Average Best Imputation Method by F1 Score Points

To get a more detailed impression regarding the significance of the difference between using the best imputation method per data constellation or just simply using K-NN,

the following Figure 7.23 illustrates all imputation results, that are ranked the best for their respective data constellation, including K-NN.

As stated before, it appears there are several scenarios offering a significant potential for improvement, relative to the average best imputation method for their respective scenarios. The Figure 7.24 shows, that the simpler imputation methods are certainly competitive in comparison with the more complex deep learning imputation approaches, especially for higher improvements in the predictive performance of the downstream machine learning models.

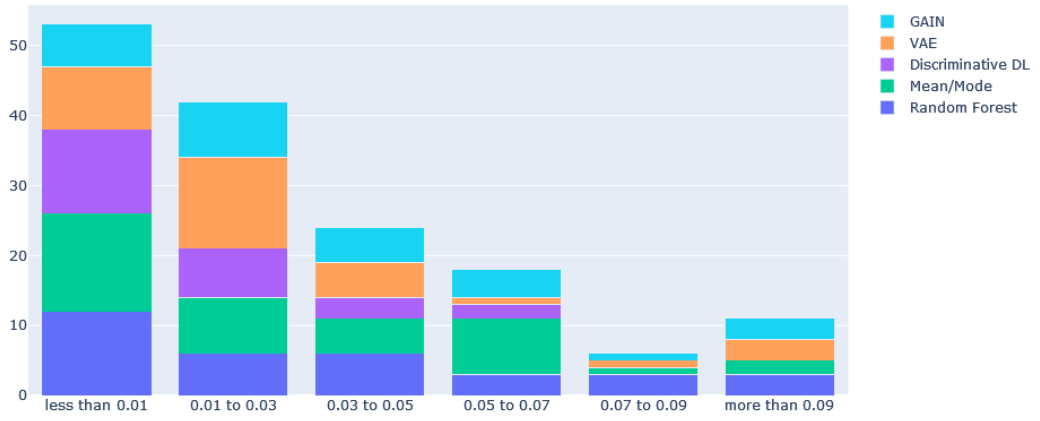

Figure 7.24.: Improvement for Best Imputation Method per Experiment Scenario Relative to the Average Best Imputation Method by F1 Score Points by Imputation Method

As described before, smaller missingness patterns achieve the majority of the more severe improvements beyond 0.09 F1 score points. Another observation from the Figure 7.25 reveals, that significant improvements are rather difficult to achieve for the smaller data subsets with the MNAR missingness pattern.

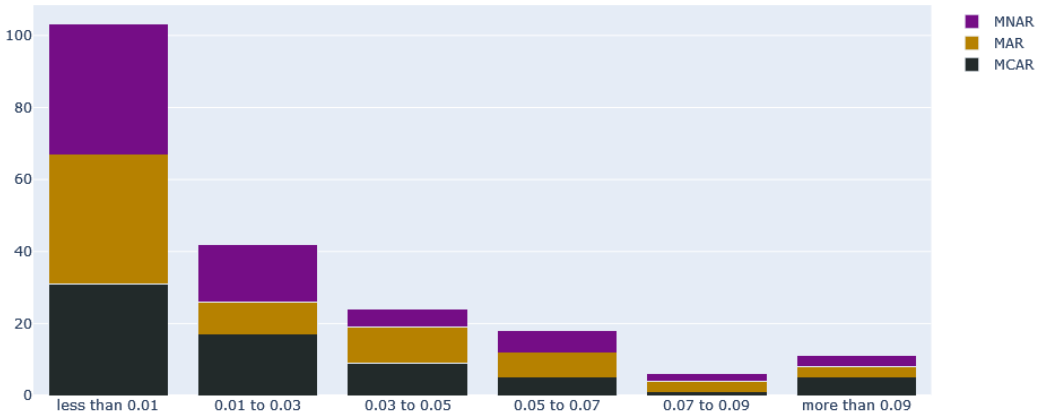

Figure 7.25.: Improvement for Best Imputation Method per Experiment Scenario Relative to the Average Best Imputation Method by F1 Score Points by Missingness Pattern

7.2.3. Regression

The Table 7.6 below shows the results for the regression experiments on the data subsets. As for the experiments on the full data set, random forest achieves on average the best rank with an average ranking of 3.21. Aside from the GAIN imputation method, all remaining five imputation approaches are again rather close in terms of average assigned rank. Despite only attaining 31 number one rankings, discriminative deep learning still reaches on average the second-best rank. This indicates a high consistency throughout the experiments. Random forest achieves 47 number one rankings but is closely followed by VAE, which despite its 45 number one rankings only achieves the fifth best rank on average. A reason for this development is the high fluctuation in terms of the outcomes for VAE, which also achieves the fifth and sixth ranking even more often the first position.

Imputation Method	Average Rank	Rank 1 Occurrences
Mean/Mode	3.4824	40
K-NN	3.4561	34
Random Forest	3.2149	47
Discriminative DL	3.3464	31
VAE	3.5964	45
GAIN	3.9035	31

Table 7.6.: Imputation Results Overview for Regression (Subset)

The difference in the absolute RMSE score from the best imputation method per data constellation and the average best imputation method for regression on data subsets (random forest) is on average 0.12. This amounts effectively to an average improvement of 0.22%, if the best imputation method is chosen instead of the average best method.

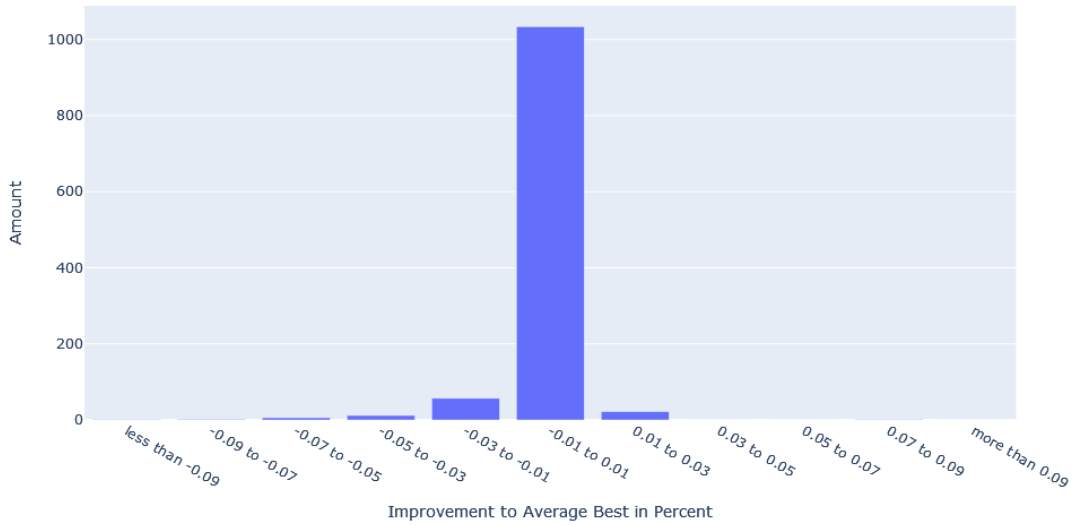

Figure 7.26.: Improvement for Imputation Experiments Relative to the Average Best Imputation Method in Percent

For further insights, a closer look at the distribution of the different results from the analysis, relative to the average best imputation method (random forest), can be undertaken. The following Figure 7.26 depicts this distribution, excluding the random forest experiments as the average best method. As seen before for the experiments on the complete datasets, it shows that the differences between the six imputation methods are rather small, given that again more than 90% of the

subset experiments with imputation methods other than random forest are achieving results within 1% improvement and 1% deterioration relative to their respective random forest result. In total, only one experiment result exceeds the improvement threshold relative to the average best method by more than 3%. While there are in total 21 experiments that resulted in a deterioration of more than 3%, these amount to less than 2% of all experiments in this section.

Like in the previous sections, to get more detailed insights regarding the importance of the difference between using the best imputation method per data constellation or just simply using random forest, the following Figure 7.27 illustrates all imputation results, that are ranked the best for their respective data constellation, including random forest. This bar chart confirms again the results described before in this section, meaning there are barely any notable improvements in downstream prediction performance for the other imputation methods in comparison to the random forest approach.

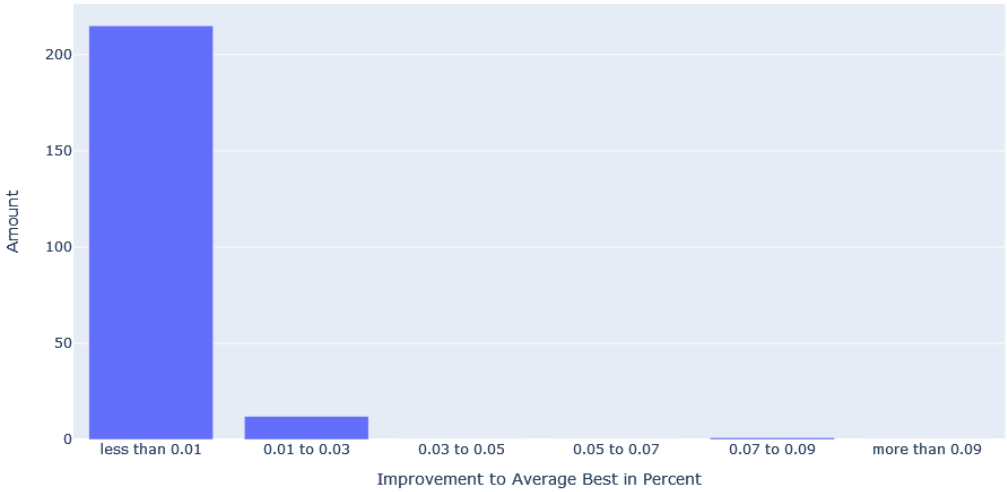

Figure 7.27.: Improvement for Best Imputation Method per Experiment Scenario Relative to the Average Best Imputation Method in Percent

7.3. Comparison of Performance Differences

This section outlines the differences between the results of the imputation experiments on the complete datasets (referred to as imputation in this section) and the experiments on the subsets of these datasets (referred to as subset in this section). In order to get a better understanding of how good or bad the predictive performance

of the downstream models is in actuality, the results of the imputation experiments on complete datasets will be compared to the performance of machine learning models, which are trained and tested on the fully observed datasets, without any corruption (referred to as baseline in this section). This gives an impression, of whether imputed training data generally improves or worsens the performance of a machine learning model. As explained in the chapter before, for classification tasks the score in the figures will be depicted in F1 score points, for regression tasks it will be in percentage.

The bar charts shown in the figures can be described as follows. On the right side of the bar chart, it shows how often the baseline model outperforms the model trained on imputed data, split into several ranges. The left side of the bar chart depicts how often the model trained on imputed data outperformed the baseline model and by how many F1 score points. These graphs include all experiments for the respective section they are in.

7.3.1. Binary Classification - Baseline to Imputation

On average, the predictive performance for the baseline models trained with uncorrupted data outscores their respective counterparts in the imputation experiments by 0.009 F1 score points.

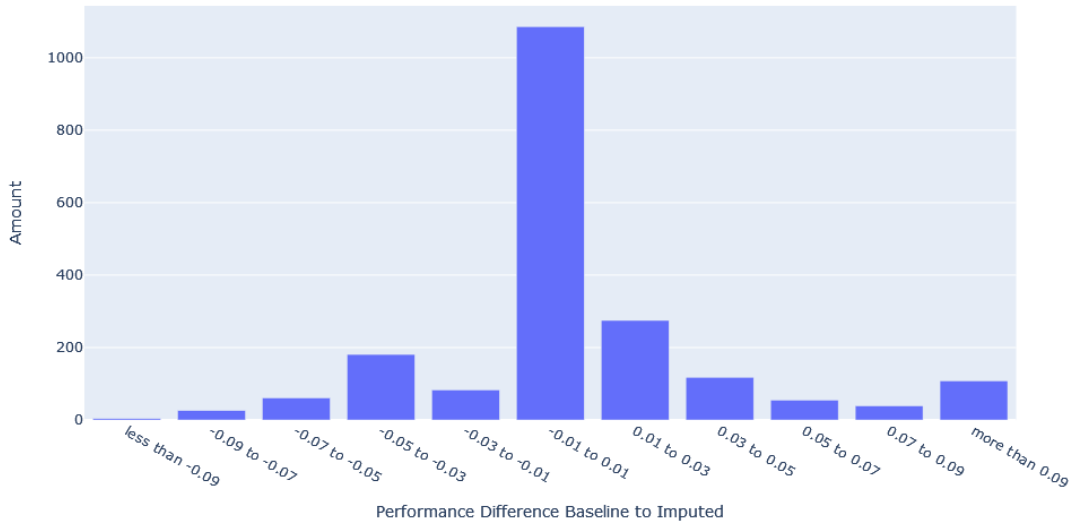

Figure 7.28.: Improvements for the Baseline Model Relative to the Model Trained on Imputed Data

The actual difference between the two results in absolute values lies on average around 0.026 F1 score points. Since there are scenarios, in which the model trained on imputed data outperforms the baseline model, the lower value for the direct comparison looks reasonable. The performance differences are portrayed in the Figure 7.28.

While there are in total more scenarios in which the baseline model outperforms the model trained on imputed data, there are still more than 300 occasions in which the baseline model gets outscored. An interesting observation is that more than half of the comparisons reveal that the differences in performance between those two models are within 0.01 F1 score points of improvement/deterioration. Further analysis reveals no clear trends for missingness pattern or imputation method since they are all rather equally represented for improvements and deteriorations, but it appears that the baseline model tends to outperform the model trained on imputed data more often for scenarios with a high fraction of corrupted data, as it can be seen in the following Figure 7.29.

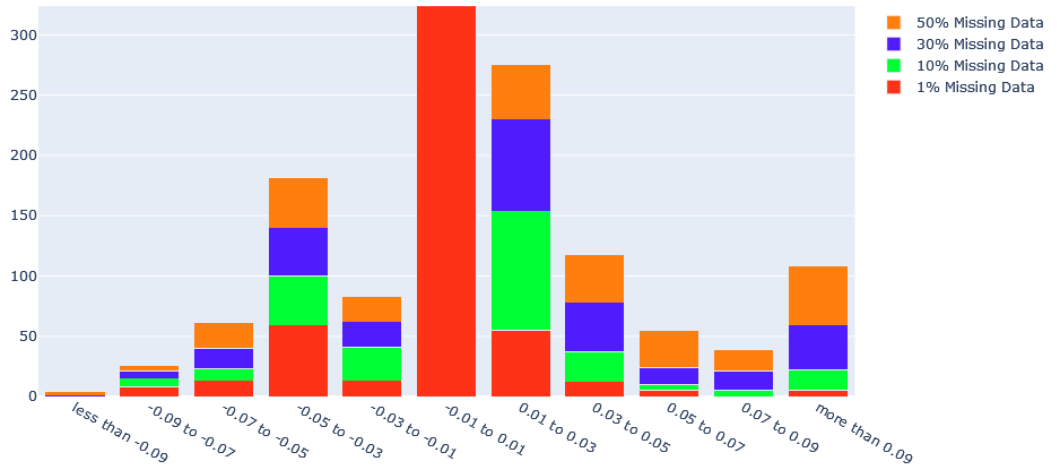

Figure 7.29.: Improvements for the Baseline Model Relative to the Model Trained on Imputed Data by Missing Fraction (Zoomed)

7.3.2. Binary Classification – Imputation to Subset

This section takes a closer look at the differences in the predictive performance of the downstream machine learning tasks, that are trained on the complete, imputed datasets and on imputed subsets of those datasets.

On average, the predictive performance for models trained on the complete, imputed datasets outscore their respective counterparts in the subset experiments by 0.008 F1 score points. While this doesn't seem much, it has to be taken into consideration, that there are experiment settings in which either model type outperforms the other. The actual difference between the two results in absolute values lies on average around 0.038 F1 score points.

The following Figure 7.30 shows the distribution of improvement and deterioration of the imputation models performance trained on the complete datasets relative to their counterparts in the subset experiments. It depicts, that the latter experiments are more often outscored, although almost 30% of models trained on a subset of their respective dataset are outscoring their counterparts trained on the full dataset by at least 0.03 F1 score points.

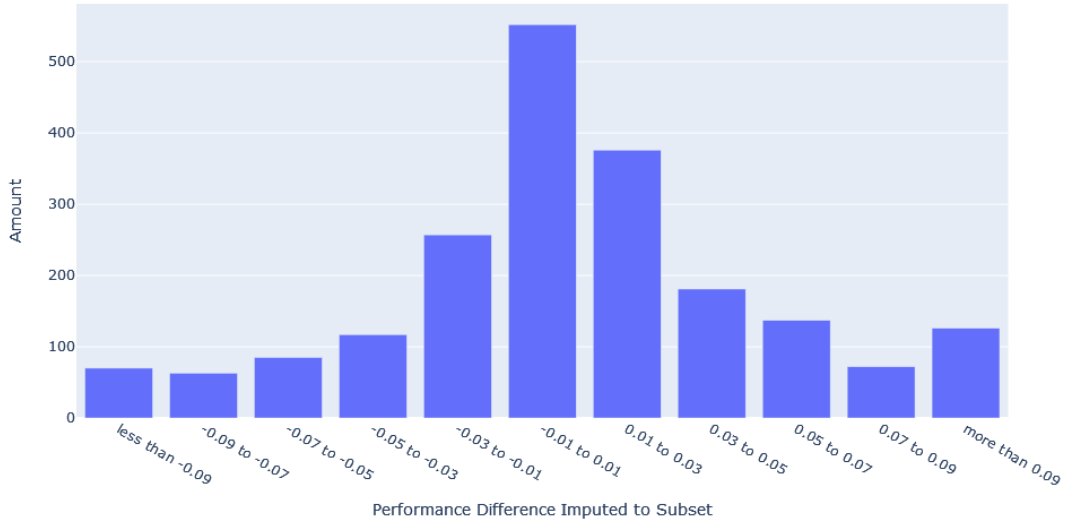

Figure 7.30.: Improvements for the Model Trained on Imputed Data Relative to the Model Trained on Subset Data

Further analysis regarding the distribution of the imputation methods, missing fractions, and missingness patterns reveals no clear favorite in either direction, rather they are remarkably well balanced throughout, as depicted in the following Figure 7.31 for the imputation methods.

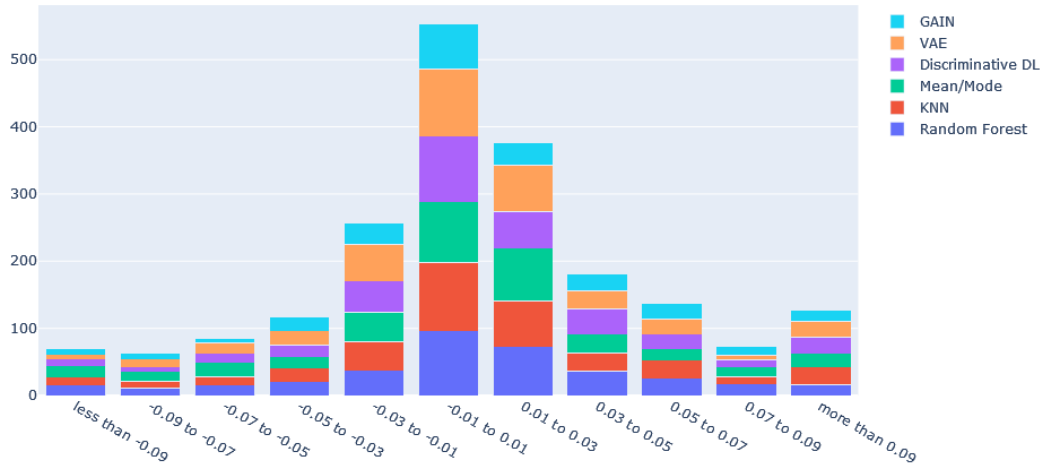

Figure 7.31.: Improvements for the Model Trained on Imputed Data Relative to the Model Trained on Subset Data by Imputation Method

An additional analysis takes a closer look at whether the subset experiments identify the same imputation methods as the best possible approach for each particular data constellation as the imputation experiments. For the binary classification almost 25% of the results are identified correctly by the subset experiments, further decoded in their respective imputation methods in the following Table 7.7. The high numbers for random forest and mean/mode can be explained by the high amount of number one rankings for both approaches in either the imputation or subset experiments.

Imputation Method	Correctly Identified	Falsely Identified
Mean/Mode	31	34
K-NN	11	49
Random Forest	21	68
Discriminative DL	8	40
VAE	6	46
GAIN	9	29

Table 7.7.: Amount of Correctly Identified Best Imputation Approaches on the Subset, split by Imputation Methods

7.3.3. Multiclass Classification - Baseline to Imputation

On average, the predictive performance for the baseline models trained with uncorrupted data outscores their respective counterparts in the imputation experiments by 0.011 F1 score points. The actual difference between the two results in absolute values lies on average around 0.022 F1 score points. The lower value for the direct comparison can be explained since there are scenarios, in which the model trained on imputed data outperforms the baseline model.

For the multiclass classification, it is more apparent that the models trained on imputed data suffer in terms of predictive performance relative to the baseline models.

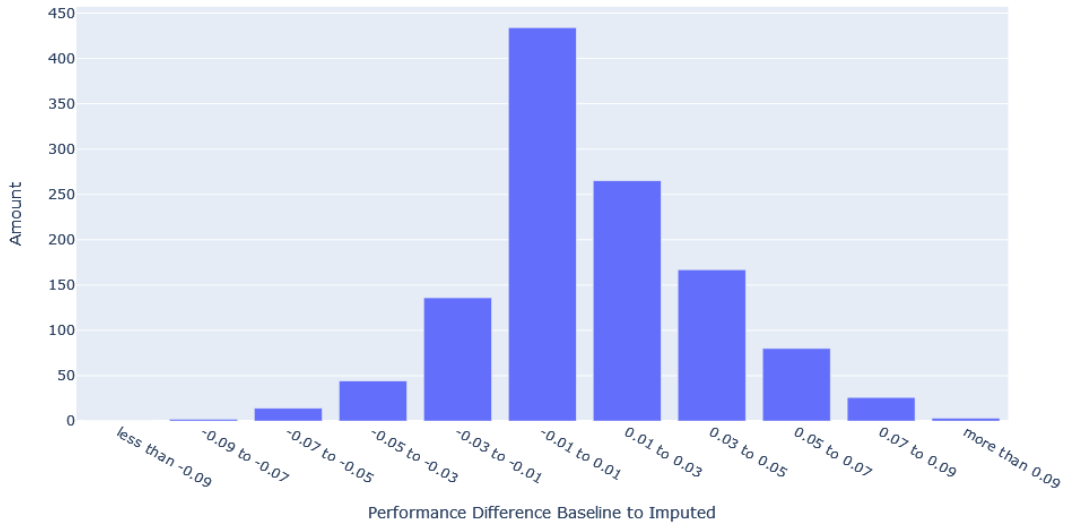

Figure 7.32.: Improvements for the Baseline Model Relative to the Model Trained on Imputed Data

As depicted in the Figure 7.32 above, the majority of results indicate that the baseline models usually outperform the models trained on imputed data, although there are still around 180 experiments, which went in the opposite direction. While there is again a clear trend, which indicates that for bigger improvements or deteriorations a higher fraction of perturbed data is involved, the distribution among the imputation methods appears again reasonably balanced, as portrayed in the Figure 7.33 below.

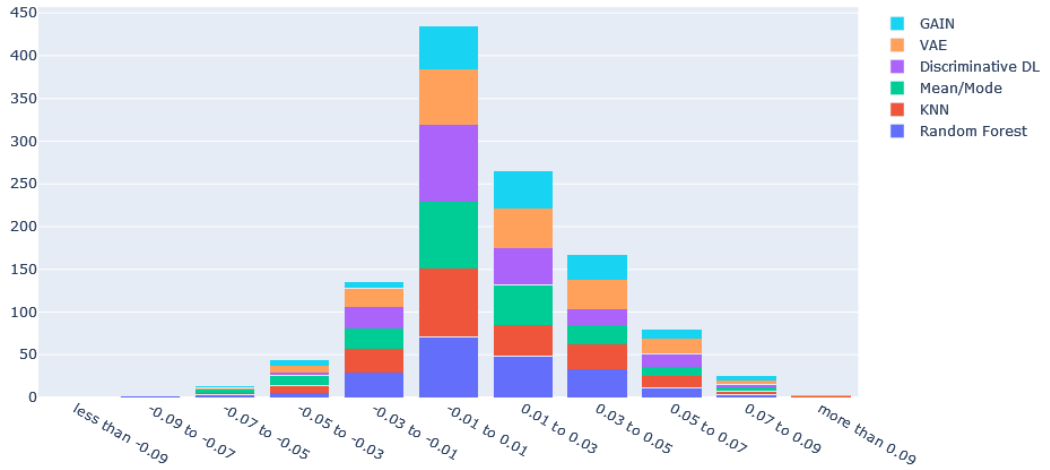

Figure 7.33.: Improvements for the Baseline Model Relative to the Model Trained on Imputed Data by Imputation Data

7.3.4. Multiclass Classification - Imputation to Subset

This section takes a closer look at the differences in the predictive performance of the downstream machine learning tasks, that are trained on the complete, imputed datasets and on subsets of those datasets.

On average, the predictive performance for models trained on the complete, imputed datasets outscore their respective counterparts in the subset experiments by 0.0028 F1 score points. It has to be considered, that there are experiment settings in which either model type outperforms the other. The actual difference between the two results in absolute values lies on average around 0.081 F1 score points. This indicates a quite significant discrepancy between the absolute results of the two experiment types.

The following Figure 7.34 shows the distribution as in previous sections. In contrast to the experiments on binary classification, it appears that more than 200 experiments scenarios with subset training data yield results with more than 0.09 F1 score points deterioration. Despite that, it can also be observed, that almost 40% of the subset experiments outscore their regular imputation experiment counterpart by more than 0.03 F1 score points.

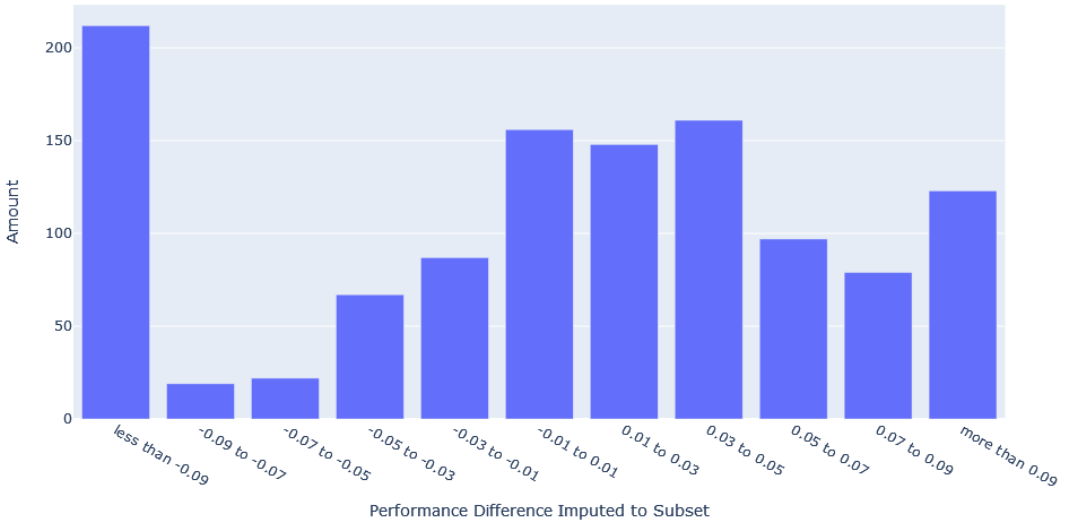

Figure 7.34.: Improvements for the Model Trained on Imputed Data Relative to the Model Trained on Subset Data

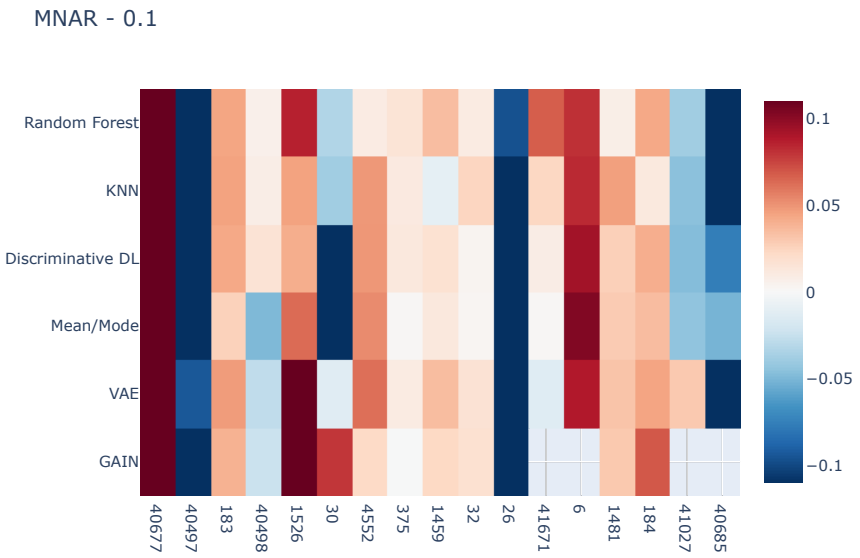

Figure 7.35.: Improvements for Models Trained on Imputed Data Relative to Models Trained on Subset Data per Dataset (MNAR and 0.1 Missing Fraction)

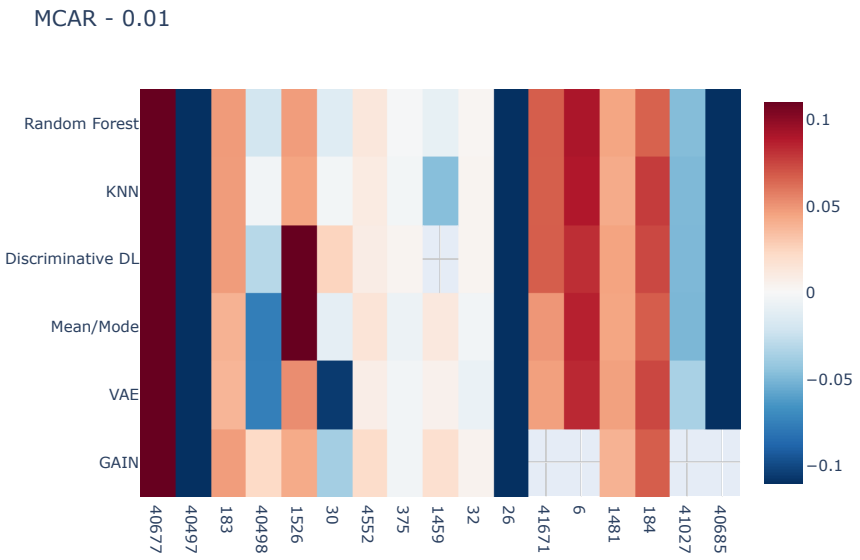

Figure 7.36.: Improvements for Models Trained on Imputed Data Relative to Models Trained on Subset Data per Dataset (MCAR and 0.01 Missing Fraction)

A closer look at the heatmaps in Figure 7.35 and Figure 7.36 reveals that this stark contrast originates mainly from a consistent over- or underperformance of the subset experiment on certain datasets. It shows the results for the experiments with the MNAR missingness pattern and a missing fraction of 10%, as well as results for the experiments with the MCAR missingness pattern and a missing fraction of 1%. The red cells indicate a better performance of the models trained on the full dataset, blue cells indicate a better performance for the subset experiment.

Further analysis reveals no clear tendency in terms of significant improvements or deteriorations based on the imputation method, missingness pattern, or missing fraction, as exemplary depicted in the following Figure 7.37 for missing fractions.

An additional analysis takes a closer look at whether the subset experiments identify the same imputation methods as the best possible approach for each particular data constellation. For the multiclass classification almost 20% of the results are identified correctly by the subset experiments, further decoded in their respective imputation methods in the following Table 7.8. The differences in the results, explained in the previous chapters, already implied that the result would be rather different. Among the imputation methods, there is no clear trend visible, nor for the other settings of the experiments.

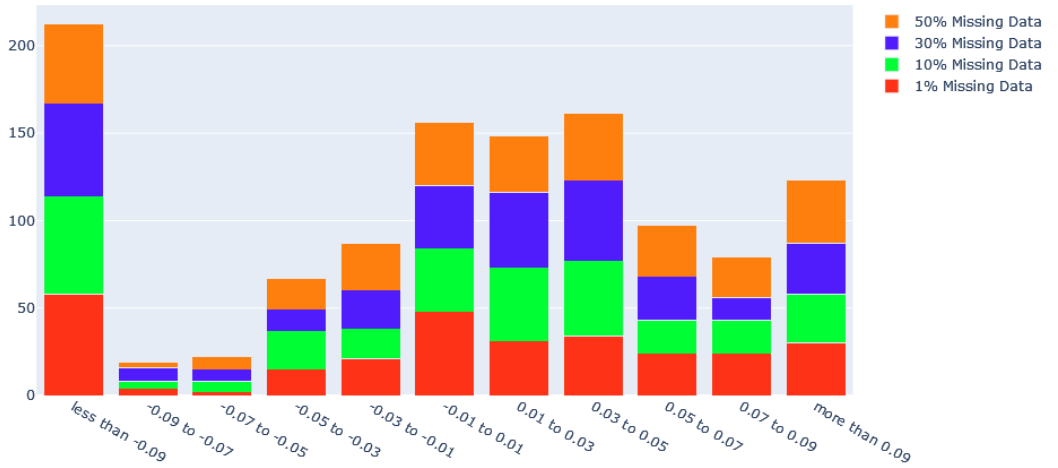

Figure 7.37.: Improvements for the Model Trained on Imputed Data Relative to the Model Trained on Subset Data by Missing Fraction

Imputation Method	Correctly Identified	Falsely Identified
Mean/Mode	7	28
K-NN	9	27
Random Forest	6	28
Discriminative DL	5	18
VAE	7	34
GAIN	5	25

Table 7.8.: Amount of Correctly Identified Best Imputation Approaches on the Subset, split by Imputation Methods

7.3.5. Regression - Baseline to Imputation

On average, the predictive performance for the baseline models trained on uncorrupted data outscores their respective counterparts in the imputation experiments for regression by 0.6%. The actual difference between the two results in absolute values lies on average around 0.72%. The lower value for the direct comparison can be explained since there are scenarios, in which the model trained on imputed data outperforms the baseline model.

In contrast to the classification tasks and in line with the just described performance

differences, for the regression tasks it appears to be rather rare that the models trained on imputed data, improve in performance. The following Figure 7.38 shows that there are in fact only 24 scenarios in which the baseline model is outperformed by more than 1%.

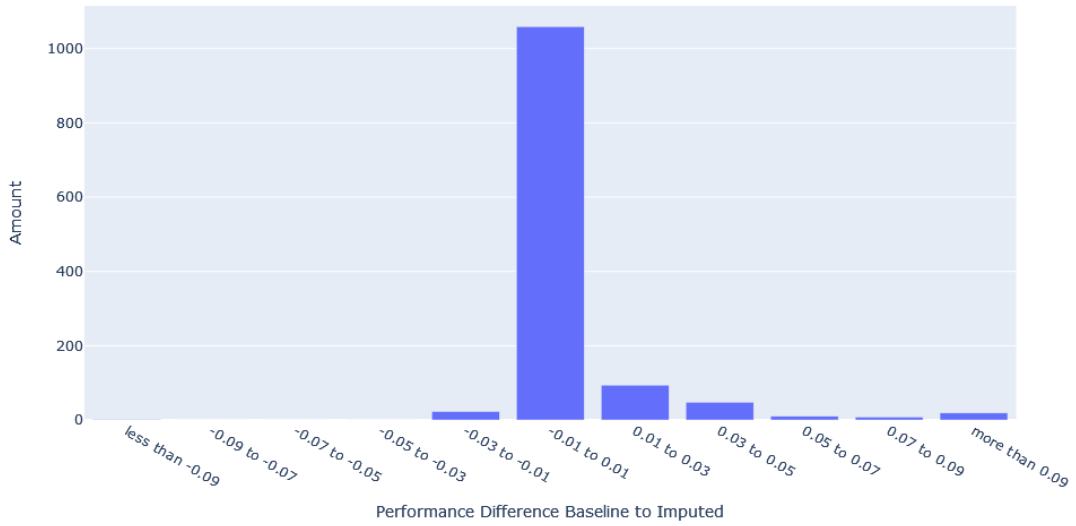

Figure 7.38.: Improvements for the Baseline Model Relative to the Model Trained on Imputed Data

This indicates that imputed training data appears to have a tendency to deteriorate the performance of regression models, preferably for scenarios in which a higher fraction of the data is imputed, as shown in the following Figure 7.39.

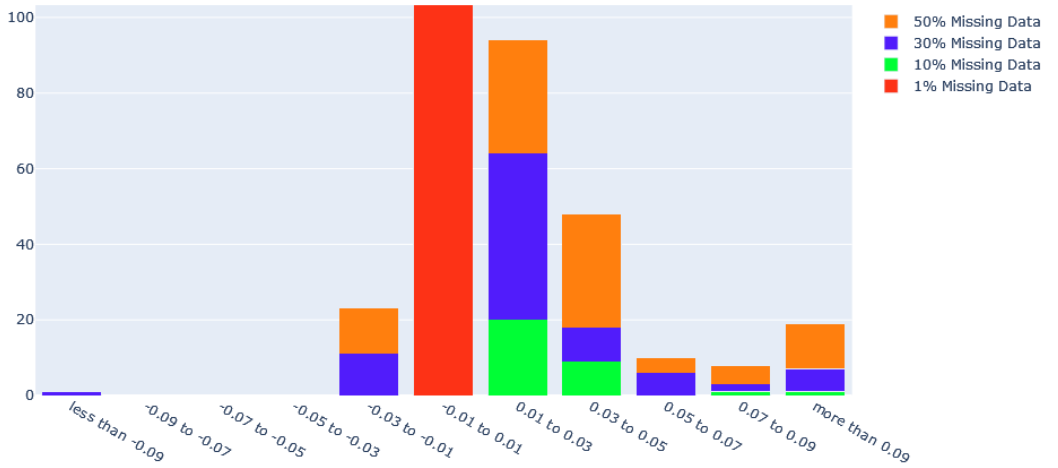

Figure 7.39.: Improvements for the Baseline Model Relative to the Model Trained on Imputed Data by Missing Fraction (Zoomed)

Nevertheless, almost 84% of results comparisons in this area are within 1% improvement or deterioration, revealing that overall the imputed training data performs relatively equal compared to the uncorrupted training data.

7.3.6. Regression - Imputation to Subset

This section takes a closer look at the differences in the predictive performance of the downstream machine learning tasks, that are trained on the complete, imputed datasets and on subsets of those datasets.

On average, the predictive performance for models trained on the complete, imputed datasets outscores their respective counterparts in the subset experiments by 48.5%. The actual difference between the two results in absolute values lies on average around 54.7%. The numbers show very clearly, that the performance of a regression machine learning model suffers severely in its predictive performance if it is not trained with a sufficient amount of data.

These numbers are visualized in the previously described manner in the following Figure 7.40. More than 63% of experiment settings reveal a significant improvement of more than 9% for the models trained on complete, imputed data over the subset training data. This underlines even further that regression models suffer severely in performance if they are trained with smaller datasets.

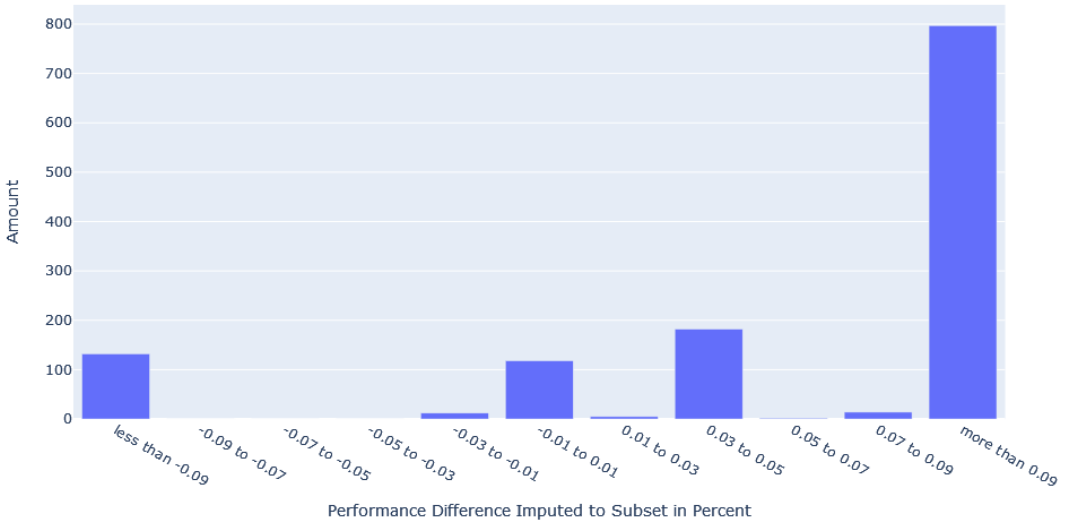

Figure 7.40.: Improvements for the Model Trained on Imputed Data Relative to the Model Trained on Subset Data

While there is no clear trend amongst the imputation method, missingness pattern, or missingness fraction visible, the following heatmaps reveal that the performance highly depends on the dataset and the selected, to-be-imputed features. As an example of this, a comparison between different experiment settings results in the following three figures (Figure 7.41; Figure 7.42; Figure 7.43). The red cells indicate a better performance of the models trained on the complete, imputed dataset, blue cells indicate better performance for the subset experiment. This illustrates the similarities between the results for the same datasets, regardless of missingness pattern or missing fraction.

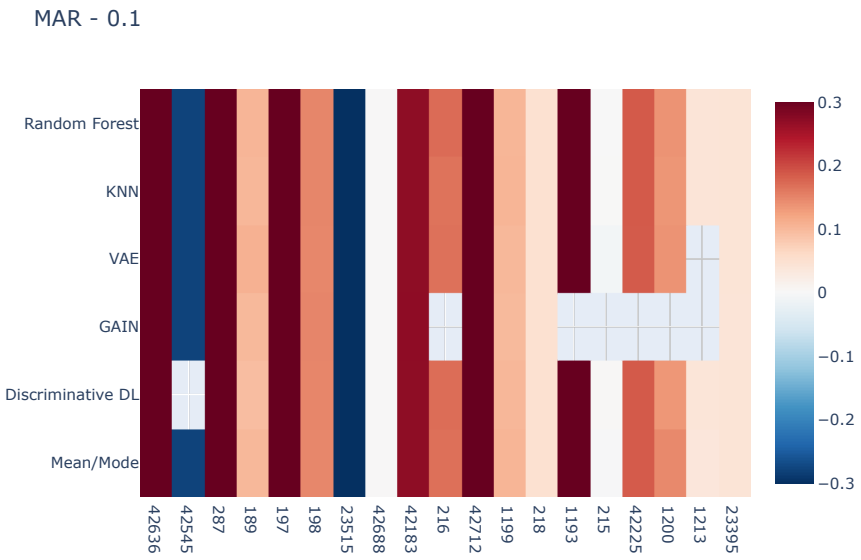

Figure 7.41.: Improvements for Models Trained on Imputed Data Relative to Models Trained on Subset Data per Dataset (MAR and 0.1 Missing Fraction)

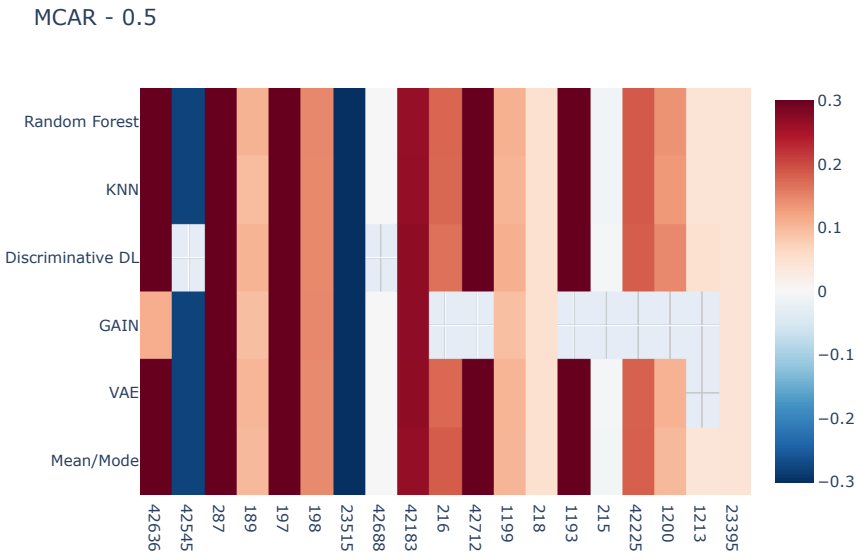

Figure 7.42.: Improvements for Models Trained on Imputed Data Relative to Models Trained on Subset Data per Dataset (MCAR and 0.5 Missing Fraction)

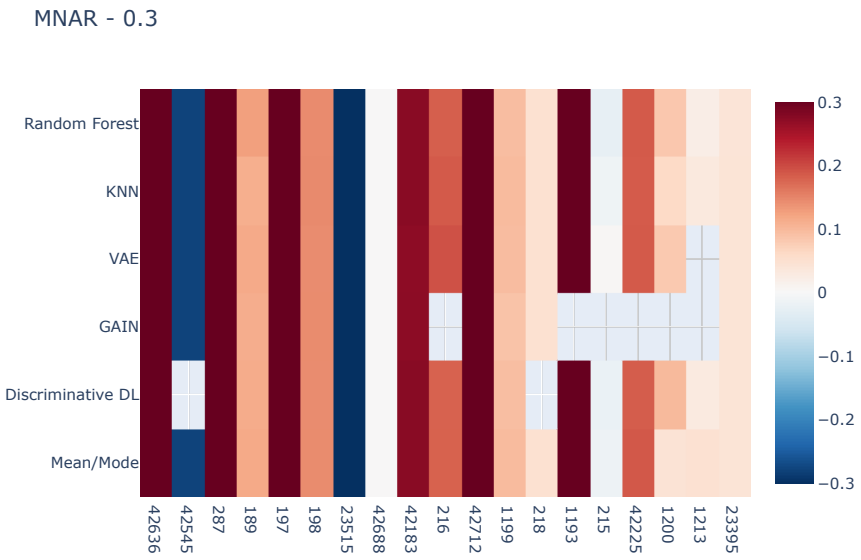

Figure 7.43.: Improvements for Models Trained on Imputed Data Relative to Models Trained on Subset Data per Dataset (MNAR and 0.3 Missing Fraction)

An additional analysis takes a closer look at whether the subset experiments identify the same imputation methods as the best possible approach for each particular data constellation. For the regression, more than 25% of the results are identified correctly by the subset experiments, further decoded in their respective imputation methods in the following Table 7.9. Due to the higher number of best rankings among random forest and mean/mode there is a higher chance, that those imputation methods are identified correctly. But aside from that, among the other imputation methods is no clear trend visible, nor for the other settings of the experiments.

Imputation Method	Correctly Identified	Falsely Identified
Mean/Mode	14	36
K-NN	3	21
Random Forest	20	28
Discriminative DL	7	29
VAE	6	23
GAIN	6	24

Table 7.9.: Amount of Correctly Identified Best Imputation Approaches on the Subset, split by Imputation Methods

7.4. Automated Recommendation Based on Dataset Properties

This chapter outlines the results and findings from the experiments to set up a classification machine learning model, which should automatically determine the best suited imputation approach for a given dataset, solely based on the dataset properties. Therefore, a random forest classifier is trained with the properties of a dataset as features and the during the imputation experiments identified, best suited imputation method as a label for each particular data constellation.

The experiments conducted for this chapter are divided in binary classification, multiclass classification, and both combined. Regression is not considered based on the results from the imputation experiments. Additionally, the model training is conducted with three different versions, in which the target label, the best suited imputation method for the particular data constellation, is adjusted. The procedure behind this versioning is explained in section 6.4.

The Table 7.10 depicts the accuracy for all versions of the input training data. Due to a ten-fold outer split in the utilized nested cross-validation, the experiments yield a total of ten different models, optimized with different hyperparameters, and tested on different test data. The first entrance per version (best) describes the best accuracy score of one of those ten models on their respective test dataset. The second entrance per version (average) shows the average accuracy for all ten models.

	Binary Classification	Multiclass Classification	Both Types of Classification
Original (best)	0.297	0.429	0.345
Original (average)	0.226	0.220	0.269
0.01 F1 Adjustment (best)	0.888	0.523	0.750
0.01 F1 Adjustment (average)	0.814	0.412	0.700
0.03 F1 Adjustment (best)	0.972	0.905	0.946
0.03 F1 Adjustment (average)	0.911	0.813	0.883

Table 7.10.: Accuracy Results for the Random Forest Model to Select the Best Suited Imputation Method

It becomes apparent, that the model is not performing sufficiently with the original dataset, containing exclusively the best imputation method per data constellation. The accuracy increases with the 0.01 and 0.03 F1 Adjustments. The reason for this increase in accuracy can be observed with a closer look at the actual predictions of the model. While the datasets become increasingly unbalanced with further adjustments, the model adapts to this, by increasing the number of random forest

suggestions. Adjusting the *class_weight* to *'balanced'*, as described in section 6.4, cannot prevent this development. This leads to situations, in which the model either only suggests random forest as the ideal solution for a given data constellation, or it suggests only one or two times a different imputation approach for a given data constellation within the test set.

This development becomes even more apparent with the following Table 7.11. In order to evaluate the quality of the suggestions of the model, the F1 scores from the imputation experiments of the suggested methods are compared to the F1 scores of random forest during the imputation experiments for those data constellations. Only the most accurate model from each experiment version from their respective nested cross-validation is considered. The results of this difference in F1 score points in depicted in Table 7.11. Due to the just described situation of simply picking random forest for most or every occasion, the improvements and deteriorations relative to random forest are negligible.

	Binary Classification	Multiclass Classification	Both Types of Classification
Original (best)	0.0041	0.0024	0.0049
1% Adjustment (best)	0.0027	-0.0054	0.0009
3% Adjustment (best)	0.0	-0.0032	0.0003

Table 7.11.: Improvement Relative to the Average Best Imputation Method (Random Forest) in F1 Score Points for the by the Model Suggested Imputation Method

These poor results can be in part traced back to the number of different data constellations for these experiments. Each data constellation represented exactly one data point for the dataset used to train the model with nested cross-validation. This amounts to 372 data points for binary classification, 204 for multiclass classification, and 576 for both combined.

7.5. Computational Complexity and Costs

In general, the target for the experiments in this thesis is to determine how data imputation for a dataset affects the training and subsequent performance of a downstream machine learning model. As the results show, in several scenarios multiple imputation methods provide almost equally good outcomes. If data scientists find themselves in a situation, where the performance aspect becomes less important due to almost or even equal outcomes, the matter of the costs to achieve these outcomes increases in importance.

In order to attain information about the computational costs, the program tracks the time it takes to fit the imputation models.

The left column of the following table shows the mean duration of the training time of the imputation model in seconds. For this analysis, all experiments are taken into consideration. The right column depicts the same information for the experiments conducted on the subsets of the data.

Imputation Method	Full Dataset Mean Duration Training	Subset Mean Duration Training
Mean/Mode	0.000476	0.000329
K-NN	1.081538	0.148897
Random Forest	17.588743	0.609447
Discriminative DL	1018.135918	198.556602
VAE	18.289359	6.183761
GAIN	182.681209	76.970171

Table 7.12.: Computational Costs for the Training of the Imputation Models in Seconds

Due to its simplicity, it is to be expected that mean/mode is by far the fastest approach. K-NN yields also a very competitive training time. While random forest and VAE require more time, they are still relatively low compared to the computational costs of GAIN and discriminative deep learning. Given the complexity and the number of optimized hyperparameters for GAIN (16 hyperparameters) and deep learning (50 hyperparameters) compared to VAE with only three hyperparameters, this outcome seems reasonable.

As mentioned in section 6.3, an important target of the subset experiments is to reduce the required training time by reducing the amount of data points. Comparing the mean training time between the experiments on the full dataset and the subset it is clear that the training time can be reduced significantly for all imputation methods.

8. Conclusions and Discussion

After investigating the impact of data imputation on the training and predictive performance of downstream machine learning models, as well as testing ways to further automate and simplify the process of picking the best imputation method, this chapter highlights and discusses the key findings of these investigations.

Several questions and hypotheses arose at the beginning of the work on this thesis and during the literature review, most of them concerning the influence and impact of imputing missing data in machine learning. While several papers, described in the literature research chapter 3, were focusing on how accurately imputation methods could predict the ground truth of missing data, this thesis aims to build on their findings, without further repeating this kind of experiment.

Since the research of Jäger et al. (2021) has the biggest influence on this thesis, the author decided to make further use of several of their experiment settings to investigate the differences between six different imputation methods in regards to their previously described impact on machine learning models. The imputation experiment results show, that random forest is on average the best imputation method for regression and both classification tasks. The different analysis results also show, that despite the fact, that random forest is not always the ideal solution, it is rarely outperformed in a significant way. In addition to that, the findings from Jäger et al. (2021) also reveal, that random forest is among the top performers in terms of imputation accuracy when predicting the ground truth of missing data. Their experiments with machine learning models trained on fully observed data and tested on imputed test data also indicate random forest as the favorite imputation method.

To further emphasize this insight, Table 8.1 depicts the differences in predictive performance relative to the random forest approach for regression (in percentage) and both classification tasks (in F1 score points). It can be seen that the baseline model, trained and tested on fully observed data, outperforms the models trained and tested in random forest imputed data. These improvements are relatively small, but they indicate a small performance advantage for models trained on complete, original data, which could be expected. The second row illustrates the average difference between the best imputation method per experiment data constellation relative to random forest. For classification, the margin for improvement is on average greater than for regression and can be quite significant in certain situations.

Based on these realizations the automated selection model, described in section 6.4, is set up to test, whether it is possible to achieve on average a better result through the suggestions from the model for the best suited imputation method for the particular data constellation at hand. This model is trained on the results of the imputation experiments and on the properties of the datasets. The model training is conducted with three different versions, in which the target label, the best suited imputation method for the particular data constellation, is adjusted, as described in section 6.4. These experiments are only conducted for the classification tasks, because the improvements for other imputation methods relative to random forest for regression are too insignificant and rare.

The results indicate that the suggestions from this model barely deviate from the F1 scores from the random forest imputation approach and therefore do not lead to a relevant improvement in performance. The poor results for these models can be in part explained by the relatively low number of different data constellations for these experiments. With 372 data constellations for binary classification, 204 data constellations for multiclass classification, and 576 data constellations for both combined, the amount of of training data for each scenario is extremely limited.

	Binary Classification	Multiclass Classification	Regression
Baseline Model	0.0024 (F1)	0.0116 (F1)	0.42 (%)
Best Imputation Method per Data Constellation	0.0108 (F1)	0.0153 (F1)	0.24 (%)
Automated Selection Model Original (best)	0.0041 (F1)	0.0024 (F1)	—
Automated Selection Model 0.01 F1 Adjustment (best)	0.0027 (F1)	-0.0054 (F1)	—
Automated Selection Model 0.03 F1 Adjustment (best)	0.0 (F1)	-0.0032 (F1)	—
Data Subet Suggestion	-0.00114 (F1)	-0.00056 (F1)	0.024 (%)

Table 8.1.: Average Improvement Relative to the Average Best Method Random Forest

There are several instances, in which the results are identical or very similar between the different imputation methods. This is often the case for smaller missing data fractions and for smaller datasets, which is in line with the findings of Woźnica and

Biecek (2020) and Zhang et al. (2018), who were not able to identify a single best imputation method in their studies, which were conducted on smaller datasets and partially on smaller missing data fractions as well.

From the observations, it can be stated that the simpler imputation methods like mean/mode, K-NN, and random forest achieve very competitive results for very low computational costs, compared to the discriminative and generative deep learning approaches, which stands in line with the findings of Poulos and Valle (2018). Nevertheless, the generative deep learning approaches, often in combination with higher fractions of imputed data, provide a higher potential for significant improvements or deteriorations for the predictive performance of the downstream machine learning model, which is trained on data imputed by these methods.

The results of the subset experiments vary, in part, greatly from the experiments on the full dataset. It is apparent, that simpler imputation methods have again the upper hand, although it often leads to a significant loss in predictive performance on the downstream machine learning model, especially for regression. With differences in the results between 0.02 and 0.38 F1 score points on average for the classification tasks and more than 50% for regression, it becomes apparent, that simply using a random subset of a dataset to determine the potential predictive performance of a machine learning model trained on the full dataset, is not a viable option.

During the subset experiments, the same ranking evaluation is conducted. For each data constellation, the subset experiments determine a best method as well. For this determined best imputation method on the subset for that particular data constellation, the author identifies the respective F1 score/RMSE for that imputation method on the complete, imputed dataset. This F1 score/RMSE is then compared to the F1 score/RMSE, the random forest approach achieved on the complete imputed dataset for that particular data constellation. Essentially it investigates the question, of whether the suggestions for the best imputation method from the subset experiments achieve on average a better F1 score/RMSE on the complete, imputed dataset, than the random forest approach does on the complete, imputed dataset. The results in the last row of Table 8.1 for this analysis show, that the suggestions from the subset experiments are achieving on average a worse F1 score than the random forest approach. For regression, there is a slight improvement visible, although an improvement of 0.024% can barely be considered worth mentioning.

To summarize, for very small datasets several imputation approaches achieve comparable results. It is in general favorable to use the random forest approach when it comes to imputing the data used to train and test downstream machine learning models. While for certain scenarios other imputation methods achieve better results, random forest usually achieves competitive results with comparably fewer resources and time effort.

9. Limitations

This chapter outlines the limitations of the research conducted in this thesis. While many aspects of this thesis boost its representative value, there are certain limitations to it.

The first important aspect to note is that for these experiments only one feature per dataset is corrupted and imputed. This feature remains static throughout the experiments. Changing this feature or adding missing data to multiple features might change the findings of this kind of research.

For reasons regarding computational cost and time, there is only one downstream model trained per experiment. The training of additional models per experiment scenario and imputation method could lead to different results as well.

To ensure that the experiments would be deterministic, certain aspects have been adjusted. All datasets have the same specific training/test split for the three repetitions. In addition to that, the exact same data points are used for all subset experiments due to the static random seed for the random selection of the data points.

Aside from those aspects, the selection criteria during the dataset selection can be seen as a limitation. While the selection of the datasets, as well as their size limits (3000 - 100000 datapoints, 5 - 25 features), exceeds what most other scientific research papers include in their research, it does not necessarily reflect more extreme real-life scenarios as seen in some industries today, where datasets with millions of data points are not considered a rarity. The datasets also only entail numerical and categorical data, image or text-based information is not considered for this thesis.

The experiments on the downstream machine learning models are also restricted to the aforementioned two types of models, regression and classification.

10. Summary and Outlook

This chapter serves to summarize the experiments and findings from this thesis, as well as to give an outlook on future developments and research around this topic.

The core topic of this thesis is to assess and predict the optimal imputation method for a given missing data scenario, keeping future usage as training data for a machine learning model in mind. This is implemented by evaluating the impact of data imputation on training and test data for the predictive performance of a downstream machine learning model, which is trained and tested on those imputed datasets. Based on this goal, six imputation models were benchmarked to test their impact on the aforementioned scenario. The random forest imputation approach proved to be on average the best imputation method for all three downstream machine learning tasks, covered in the experiments (regression, binary and multiclass classification). Since random forest does not present the best solution for every covered situation and was outperformed quite significantly multiple times by other imputation approaches, the question arose, whether it would be possible to determine the best suited imputation method for a dataset at hand in a quicker, more automated way.

Therefore, in order to reduce the time required for these experiments, the same tests were conducted on a subset of each datasets, which were chosen for these experiments. By doing so, the computational cost and processing time could be reduced significantly, but the results and subsequent suggestions for the to-be-chosen imputation methods yielded no improvement, relative to the approach, by which simply random forest is used every time.

Another approach to simplify the decision for the best suited imputation method is to train a classification machine learning model with the properties of a dataset as features and the identified, best suited imputation method as a label. Due to the lack of improvement possibilities, regression tasks were excluded from these tests. The suggestions made by the models for binary classification, multiclass classification, and both combined were unfortunately not capable of achieving an on average significantly better F1 score than the pure random forest approach.

Aside from those realizations, the thesis confirmed previous findings by other researchers, which stated that simpler imputation methods are capable of achieving competitive results relative to more complex approaches. Especially on smaller datasets, like during the subset experiments, this observation became apparent. In addition to that, it can be stated that downstream machine learning models, trained

and tested on random forest imputed data, are barely outscored by models trained and tested on fully observed and uncorrupted data.

Given that these experiments were conducted on a larger number of datasets with a considerable number of features and datapoints, the results of this thesis can be considered representative of real-life situations, within the border outlined in the previous chapter regarding the limitations of this thesis.

For future research, it might be of interest to further increase the quantity of the conducted experiments to increase their weight and representative value. Additional downstream machine learning tasks, like neural networks, could diversify this research further. An increase in the amount and size of the datasets, or the use of more imputation methods, as well as the corruption of multiple feature columns would be ways to proceed and build on the findings of this thesis.

The topic of missing data imputation will remain relevant in the future. This is further highlighted by the number of research papers published in more recent years and the ever-growing demand for more high-quality data. Artificial intelligence is becoming increasingly more engrained in today's society, starting with technologies like face and motion recognition and industrial robots, over recommendation systems in online shopping and social media, to tools for personal usage like voice recognition for personal assistants or simply the spam filter in everyone's email account.

To ensure that these products and applications perform at their best and support humans in their daily life, it is vital to ensure the best possible level of data quality, with completeness, meaning no missing data, being one of the most important aspect of this quality.

"Data is the new oil", a quote from the British mathematician Clive Humby (Viernes, 2022), is already stated in the introduction of this thesis, but in recent years it has become clear, that raw data, just like raw oil, on its own is not sufficient to efficiently fuel anything. It needs to be preprocessed in order to be facilitated. And while mankind has already several centuries worth of experience in the preprocessing of the old, original oil, it is now time, to start developing and gaining that level of knowledge for the proclaimed new oil of the 21. century.

Bibliography

2U, I. (2022, July). How to Deal with Missing Data [[Online; accessed 9. May 2023]]. https://www.mastersindatascience.org/learning/how-to-deal-with-missing-data

Aljuaid, T., & Sasi, S. (2016, August). Proper imputation techniques for missing values in data sets. In *2016 International Conference on Data Science and Engineering (ICDSE)* (pp. 1–5). IEEE. https://doi.org/10.1109/ICDSE.2016.7823957

Batista, G. E. A. P. A., & Monard, M. C. (2003). An analysis of four missing data treatment methods for supervised learning. *Applied Artificial Intelligence*, *17*(5-6), 519–533. https://doi.org/10.1080/713827181

Bertsimas, D., Pawlowski, C., & Zhuo, Y. D. (2017). From predictive methods to missing data imputation: an optimization approach. *J. Mach. Learn. Res.*, *18*(1), 7133–7171. https://doi.org/10.5555/3122009.3242053

Biessmann, F., Salinas, D., Schelter, S., Schmidt, P., & Lange, D. (2018, October). "Deep" Learning for Missing Value Imputation in Tables with Non-Numerical Data. In *CIKM '18: Proceedings of the 27th ACM International Conference on Information and Knowledge Management* (pp. 2017–2025). Association for Computing Machinery. https://doi.org/10.1145/3269206.3272005

Brocke, J. v., Hevner, A., & Maedche, A. (2020, September). Introduction to Design Science Research. In *Design Science Research. Cases*. Springer. https://doi.org/10.1007/978-3-030-46781-4_1

Camino, R. D., Hammerschmidt, C. A., & State, R. (2019). Improving Missing Data Imputation with Deep Generative Models. *arXiv*. https://doi.org/10.48550/arXiv.1902.10666

Emmanuel, T., Maupong, T., Mpoeleng, D., Semong, T., Mphago, B., & Tabona, O. (2021). A survey on missing data in machine learning. *J. Big Data*, *8*(1), 1–37. https://doi.org/10.1186/s40537-021-00516-9

García Laencina, P., Sancho-Gómez, J. L., & Figueiras-Vidal, A. (2010). Pattern classification with missing data: A review. *Neural Computing and Applications*, *19*, 263–282. https://doi.org/10.1007/s00521-009-0295-6

Gawande, S. (2021, March). *6 Dimensions of Data Quality, Examples, and Measurement* [[Online; accessed 13. Aug. 2022]]. https://icedq.com/6-data-quality-dimensions

Gregor, S., & Hevner, A. R. (2013). Positioning and Presenting Design Science Research for Maximum Impact on JSTOR. *MIS Quarterly*, *37*(2), 337–355. https://www.jstor.org/stable/43825912

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., & Hochreiter, S. (2017). Gans trained by a two time-scale update rule converge to a local nash equilibrium. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in neural information processing systems* (Vol. 30). Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2017/file/8a1d694707eb0fefe65871369074926d-Paper.pdf

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design Science in Information Systems Research on JSTOR. *MIS Quarterly*, *28*(1), 75–105. https://www.jstor.org/stable/25148625

Hub, G. D. Q. (2021). Meet the data quality dimensions. *GOV*. https://www.gov.uk/government/news/meet-the-data-quality-dimensions

Jadhav, A., Pramod, D., & Ramanathan, K. (2019). Comparison of Performance of Data Imputation Methods for Numeric Dataset. *Applied Artificial Intelligence*, *33*(10), 913–933. https://doi.org/10.1080/08839514.2019.1637138

Jäger, S., Allhorn, A., & Bießmann, F. (2021). A Benchmark for Data Imputation Methods. *Front. Big Data*, *4*. https://doi.org/10.3389/fdata.2021.693674

Jin, H., François, C., Qingquan, S., & Hu, X. (2023, January). StructuredDataRegressor by AutoKeras [[Online; accessed 25. May 2023]]. https://autokeras.com/structured_data_regressor

Jin, H., François, C., Qingquan, S., & Hu, X. (2023-01). StructuredDataClassifier by AutoKeras [[Online; accessed 25. May 2023]]. https://autokeras.com/structured_data_classifier

Jin, H., Song, Q., & Hu, X. (2019, July). Auto-Keras: An Efficient Neural Architecture Search System. In *KDD '19: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (pp. 1946–1956). Association for Computing Machinery. https://doi.org/10.1145/3292500.3330648

Miyato, T., Kataoka, T., Koyama, M., & Yoshida, Y. (2018). Spectral Normalization for Generative Adversarial Networks. *arXiv*. https://doi.org/10.48550/arXiv.1802.05957

Munson, M. A. (2012). A study on the importance of and time spent on different modeling steps. *SIGKDD Explor. Newsl.*, *13*(2), 65–71. https://doi.org/10.1145/2207243.2207253

Poulos, J., & Valle, R. (2018). Missing Data Imputation for Supervised Learning. *Applied Artificial Intelligence*, *32*(2), 186–196. https://doi.org/10.1080/08839514.2018.1448143

Reyes, K. (2023). What is Deep Learning and How Does It Works [Explained]. *Simplilearn*. https://www.simplilearn.com/tutorials/deep-learning-tutorial/what-is-deep-learning#what_is_deep_learning

Rocca, J. (2021a). Understanding Generative Adversarial Networks (GANs). *Medium*. https://towardsdatascience.com/understanding-generative-adversarial-networks-gans-cd6e4651a29

Rocca, J. (2021b). Understanding Variational Autoencoders (VAEs). *Medium*. https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

Rubin, D. B. (1976). Inference and missing data. *Biometrika*, *63*(3), 581–592. https://doi.org/10.1093/biomet/63.3.581

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X., & Chen, X. (2016). Improved techniques for training gans. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, & R. Garnett (Eds.), *Advances in neural information processing systems* (Vol. 29). Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2016/file/8a3363abe792db2d8761d6403605aeb7-Paper.pdf

Schelter, S., Rukat, T., & Biessmann, F. (2021). Jenga: A framework to study the impact of data errors on the predictions of machine learning models. *EDBT 2021 Industrial and Application Track*. https://www.amazon.science/publications/jenga-a-framework-to-study-the-impact-of-data-errors-on-the-predictions-of-machine-learning-models

sklearn RandomForestClassifier [[Online; accessed 22. May 2023]]. (2023, May). https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

sklearn SGDClassifier [[Online; accessed 19. May 2023]]. (2023, May). https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html

sklearn SGDRegressor [[Online; accessed 19. May 2023]]. (2023, May). https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDRegressor.html

Turing. (2022, April). Generative Models vs Discriminative Models: Which One to Choose for Deep Learning and Why? [[Online; accessed 6. Apr. 2023]]. https://www.turing.com/kb/generative-models-vs-discriminative-models-for-deep-learning#discriminative-vs-generative:-which-is-the-best-fit-for-deep-learning

Uppenkamp, M. (2022). Die häufigsten Datenqualitätsprobleme. *INFORM Datalab*. https://www.inform-datalab.de/die-haeufigsten-datenqualitaetsprobleme

van Buuren, S. (2018, July). *Flexible Imputation of Missing Data, Second Edition*. Taylor & Francis. https://www.taylorfrancis.com/books/edit/10.1201/9780429492259/flexible-imputation-missing-data-second-edition-%20stef-van-buuren

Viernes, F. A. (2022, January). *Stop Saying 'Data is the New Oil' - Geek Culture - Medium*. Geek Culture. https://medium.com/geekculture/stop-saying-data-is-the-new-oil-a2422727218c

What is Random Forest? by IBM [[Online; accessed 6. Apr. 2023]]. (2023, April). https://www.ibm.com/topics/random-forest

What is the k-nearest neighbors algorithm? by IBM [[Online; accessed 6. Apr. 2023]]. (2023, April). https://www.ibm.com/uk-en/topics/knn

Woźnica, K., & Biecek, P. (2020). Does imputation matter? Benchmark for predictive models. *arXiv*. https://doi.org/10.48550/arXiv.2007.02837

Wolf, Z. B. (2023). AI can be racist, sexist and creepy. What should we do about it? | CNN Politics. *CNN*. https://edition.cnn.com/2023/03/18/politics/ai-chatgpt-racist-what-matters/index.html

Yoon, J., Jordon, J., & Schaar, M. (2018, July). GAIN: Missing Data Imputation using Generative Adversarial Nets. In *International Conference on Machine Learning* (pp. 5689–5698). PMLR. http://proceedings.mlr.press/v80/yoon18a.html?ref=https://githubhelp.com

Zhang, H., Xie, P., & Xing, E. (2018). Missing Value Imputation Based on Deep Generative Models. *arXiv*. https://doi.org/10.48550/arXiv.1808.01684

A. Appendix Binary Classification Experiments

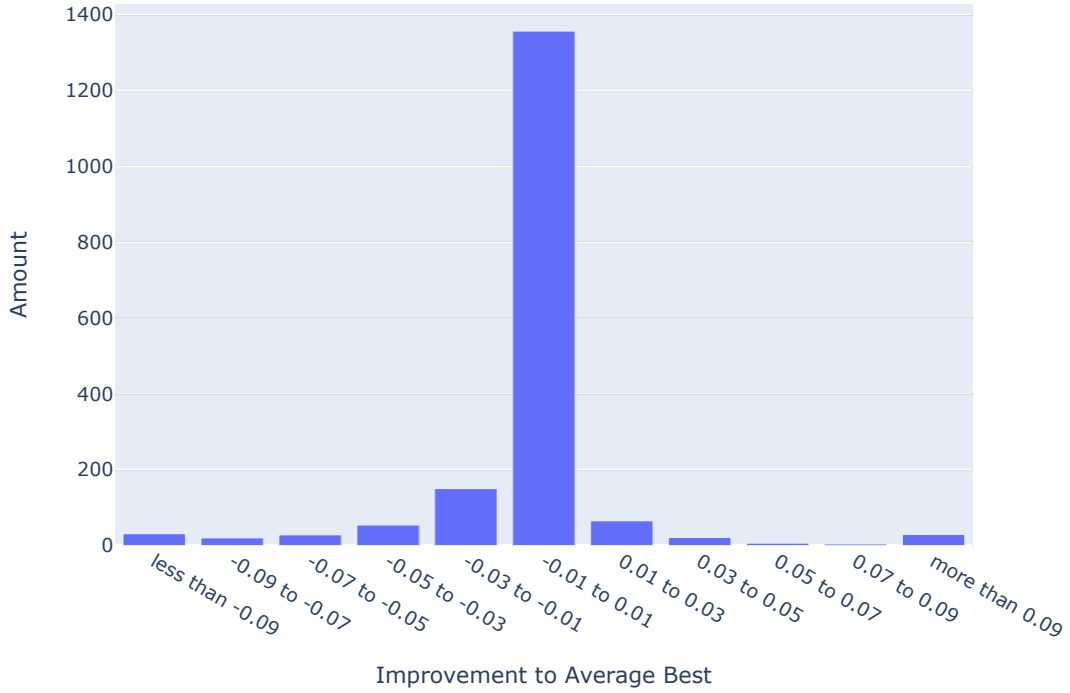

Figure A.1.: Improvement for Experiments Relative to the Average Best Imputation Method by F1 Score Points

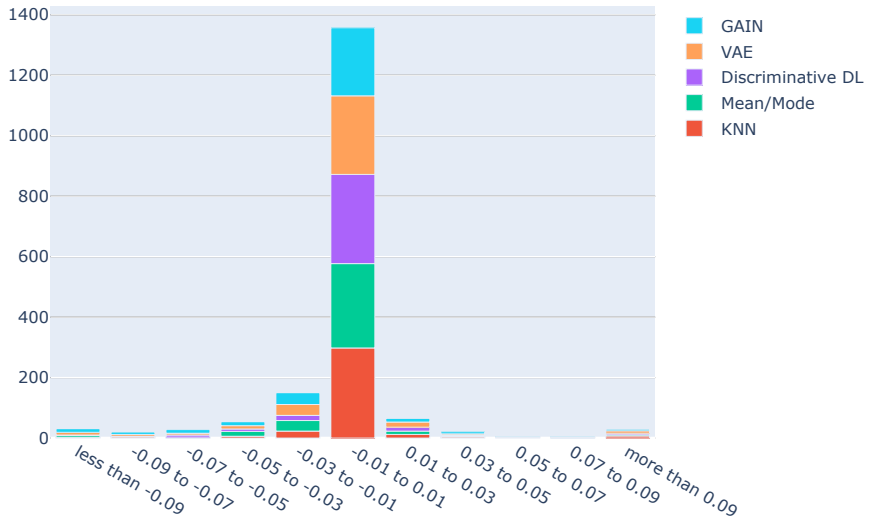

Figure A.2.: Improvement for Experiments Relative to the Average Best Imputation Method by F1 Score Points by Imputation Methods

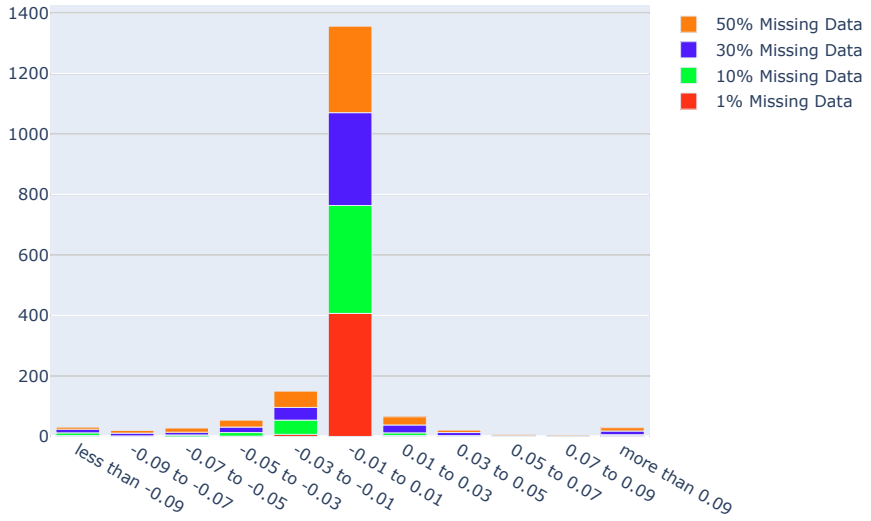

Figure A.3.: Improvement for Experiments Relative to the Average Best Imputation Method by F1 Score Points by Missing Fraction

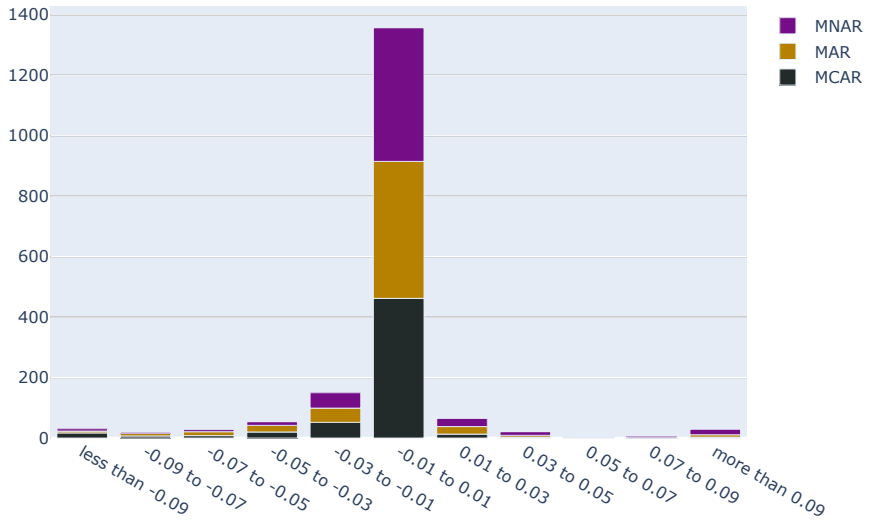

Figure A.4.: Improvement for Experiments Relative to the Average Best Imputation Method by F1 Score Points by Missingness Pattern

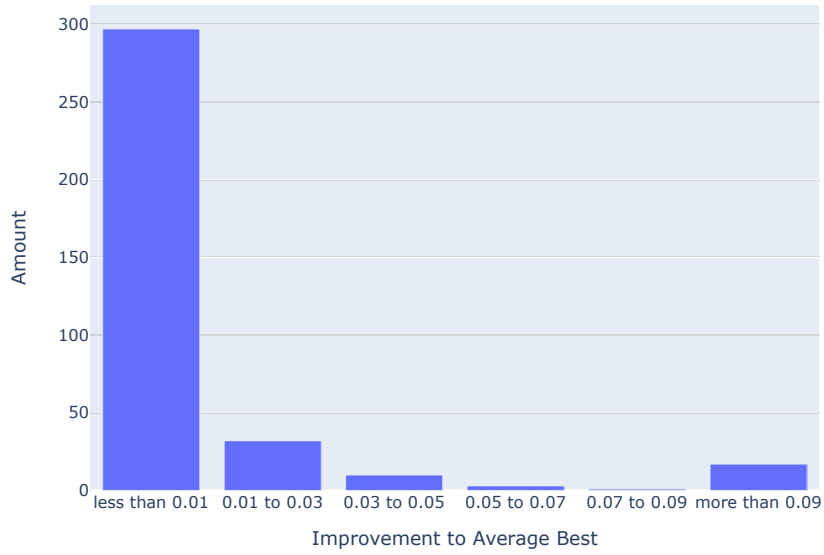

Figure A.5.: Improvement for Best Imputation Method per Experiment Scenario Relative to the Average Best Imputation Method by F1 Score Points

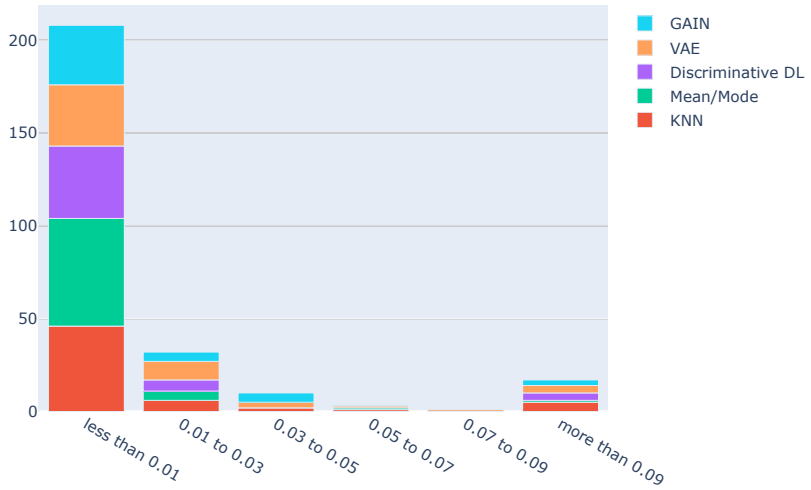

Figure A.6.: Improvement for Best Imputation Method per Experiment Scenario Relative to the Average Best Imputation Method by F1 Score Points by Imputation Method

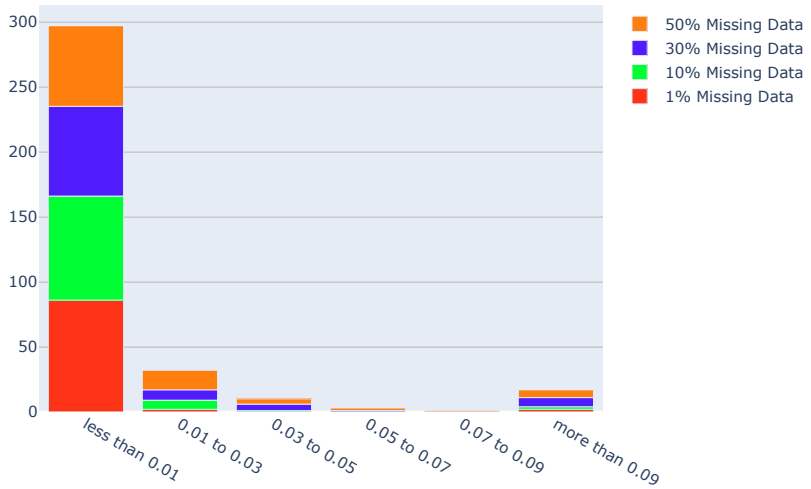

Figure A.7.: Improvement for Best Imputation Method per Experiment Scenario Relative to the Average Best Imputation Method by F1 Score Points by Missing Fraction

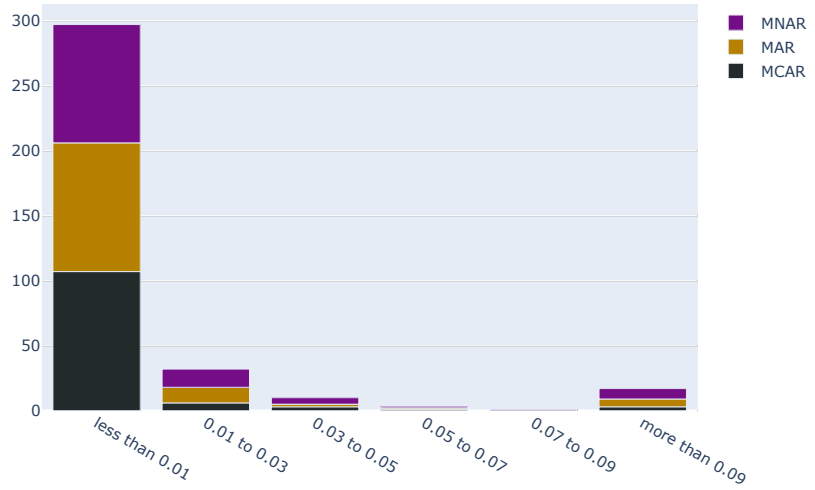

Figure A.8.: Improvement for Best Imputation Method per Experiment Scenario Relative to the Average Best Imputation Method by F1 Score Points Missingness Pattern

B. Appendix Multiclass Classification Experiments

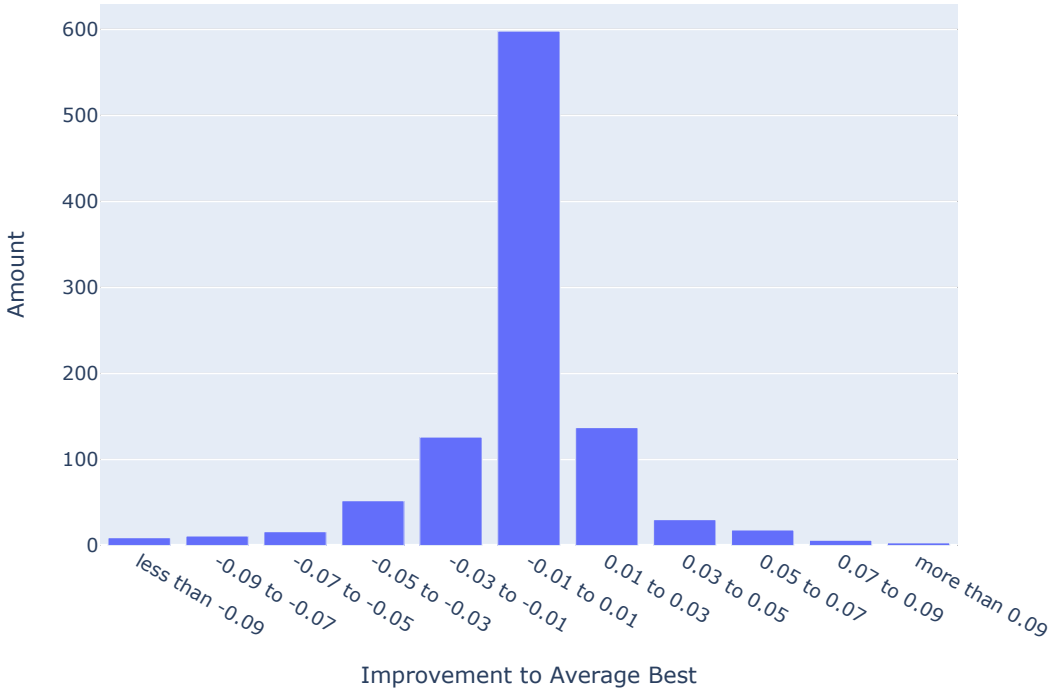

Figure B.1.: Improvement for Experiments Relative to the Average Best Imputation Method by F1 Score Points

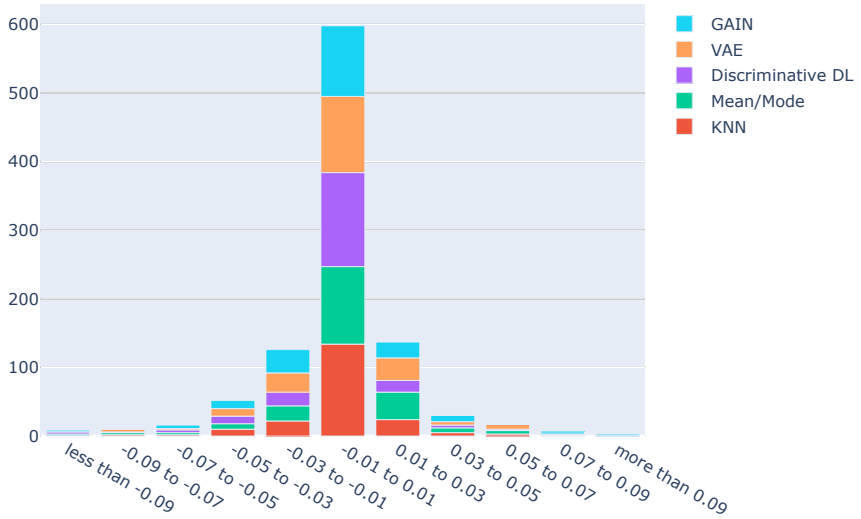

Figure B.2.: Improvement for Experiments Relative to the Average Best Imputation Method by F1 Score Points by Imputation Method

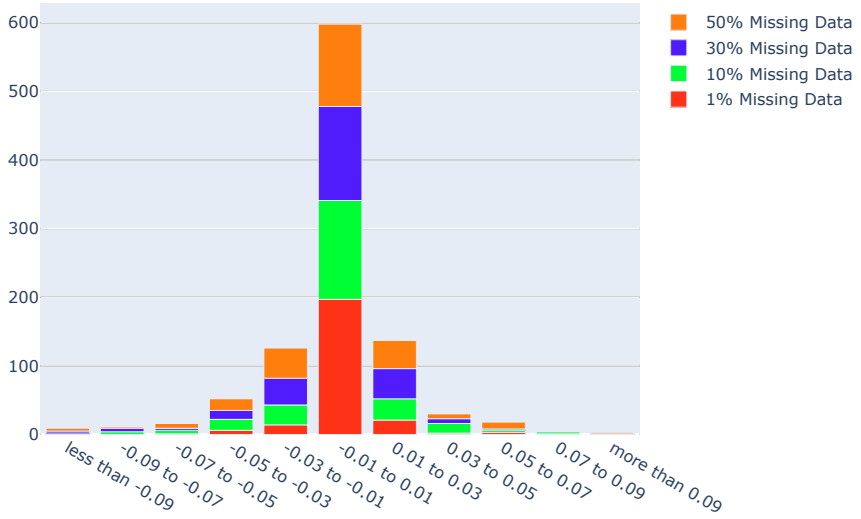

Figure B.3.: Improvement for Experiments Relative to the Average Best Imputation Method by F1 Score Points by Missing Fraction

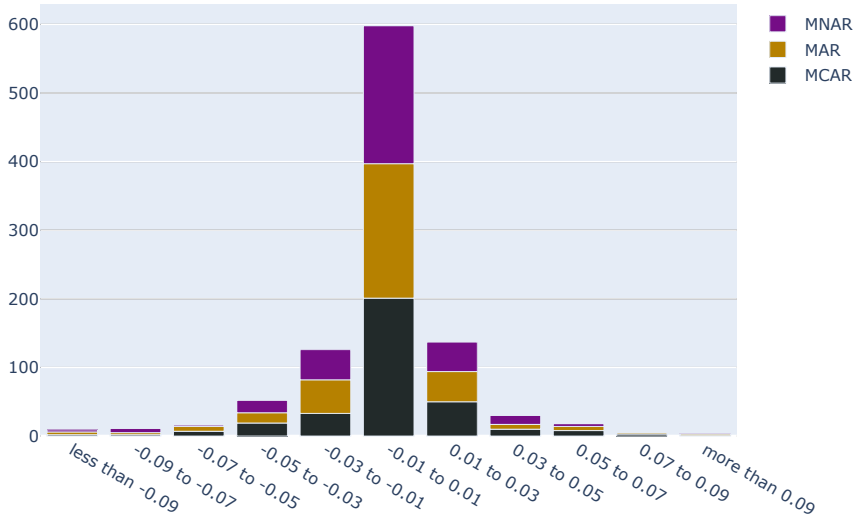

Figure B.4.: Improvement for Experiments Relative to the Average Best Imputation Method by F1 Score Points by Missingness Pattern

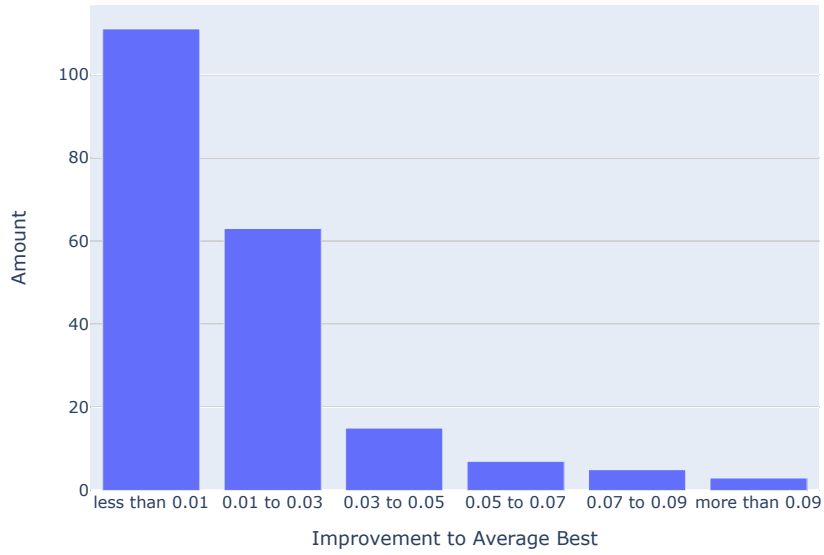

Figure B.5.: Improvement for Best Imputation Method per Experiment Scenario Relative to the Average Best Imputation Method by F1 Score Points

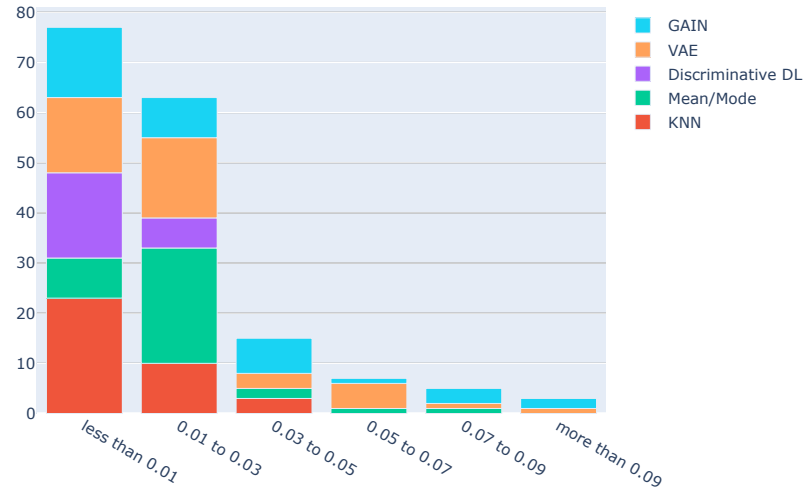

Figure B.6.: Improvement for Best Imputation Method per Experiment Scenario Relative to the Average Best Imputation Method by F1 Score Points by Imputation Method

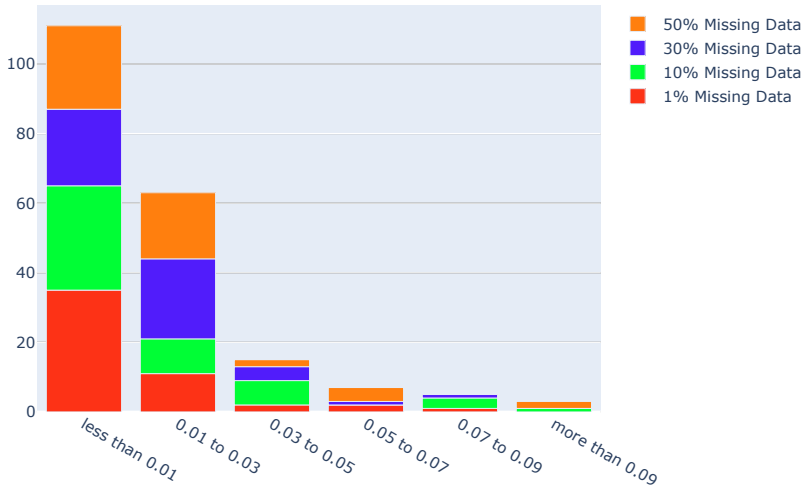

Figure B.7.: Improvement for Best Imputation Method per Experiment Scenario Relative to the Average Best Imputation Method by F1 Score Points by Missing Fraction

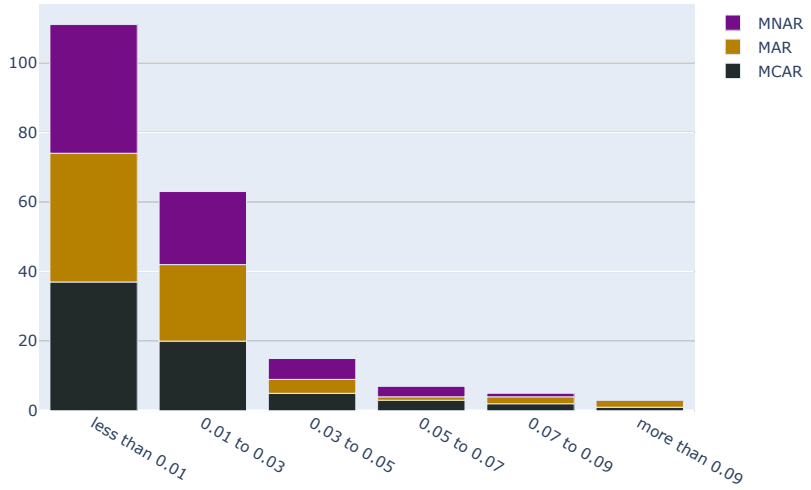

Figure B.8.: Improvement for Best Imputation Method per Experiment Scenario Relative to the Average Best Imputation Method by F1 Score Points by Missingness Pattern

C. Appendix Regression Experiments

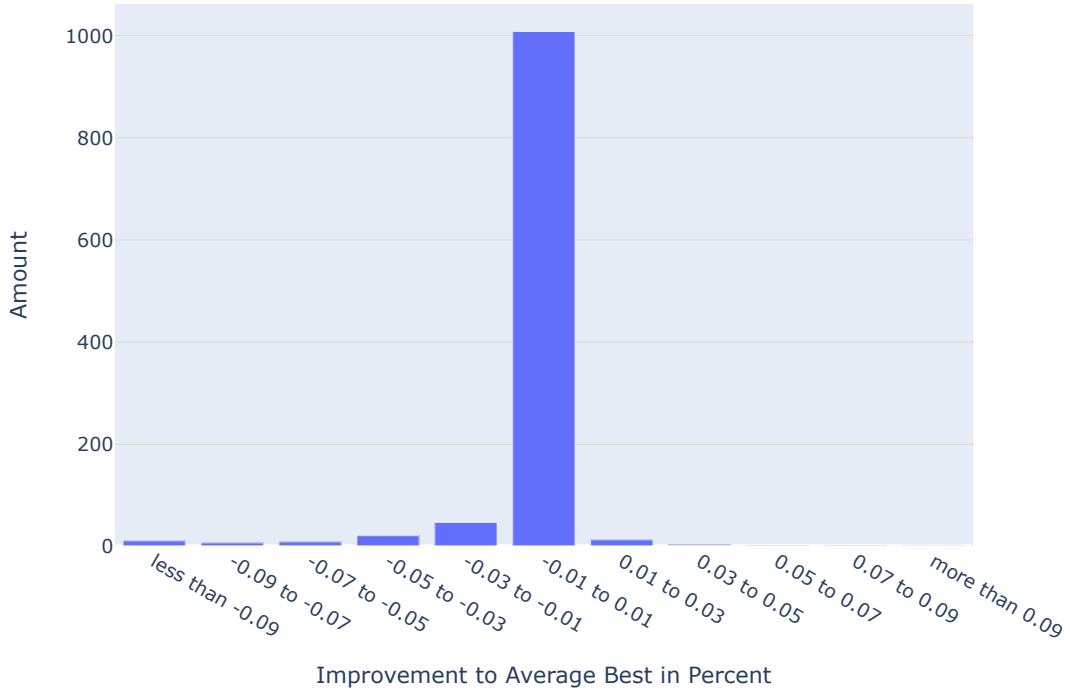

Figure C.1.: Improvement for Experiments Relative to the Average Best Imputation Method by Percentage

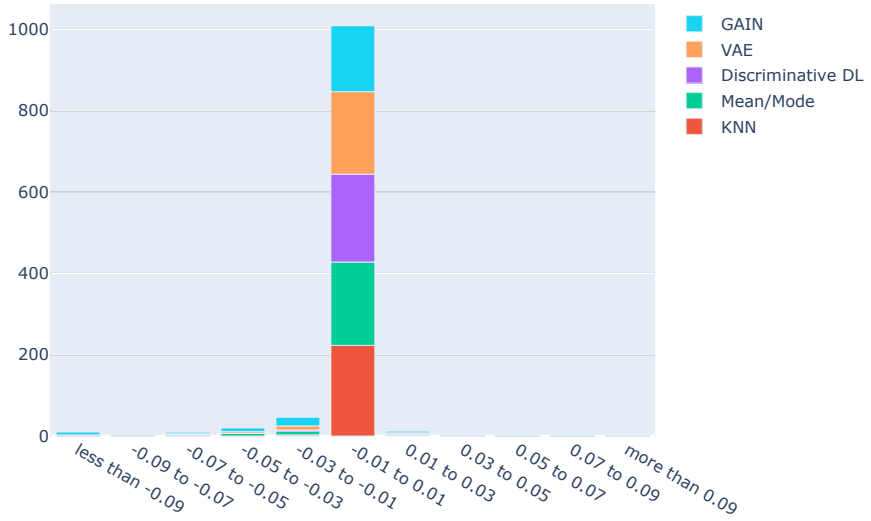

Figure C.2.: Improvement for Experiments Relative to the Average Best Imputation Method by Percentage by Imputation Method

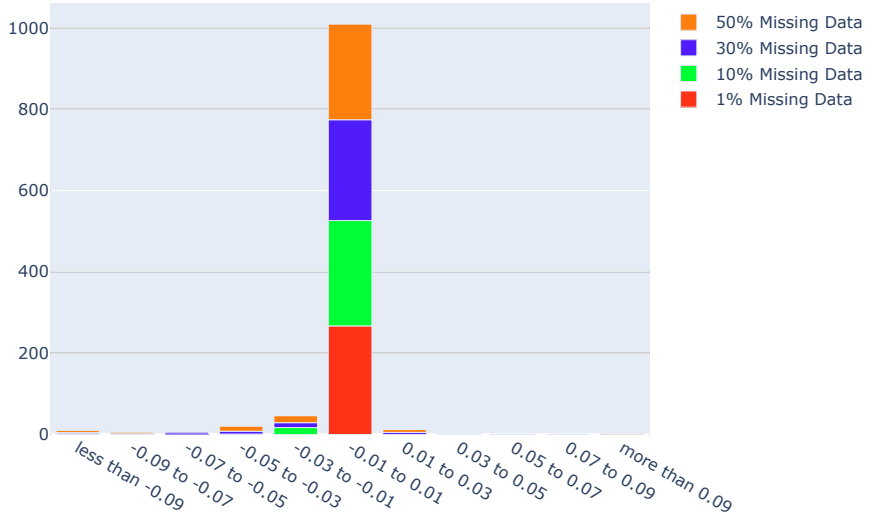

Figure C.3.: Improvement for Experiments Relative to the Average Best Imputation Method by Percentage by Missing Fraction

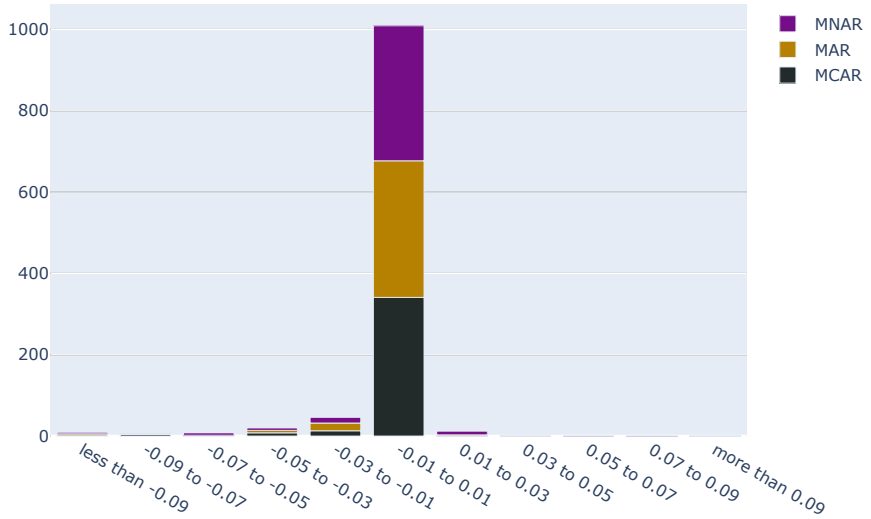

Figure C.4.: Improvement for Experiments Relative to the Average Best Imputation Method by Percentage by Missingness Pattern

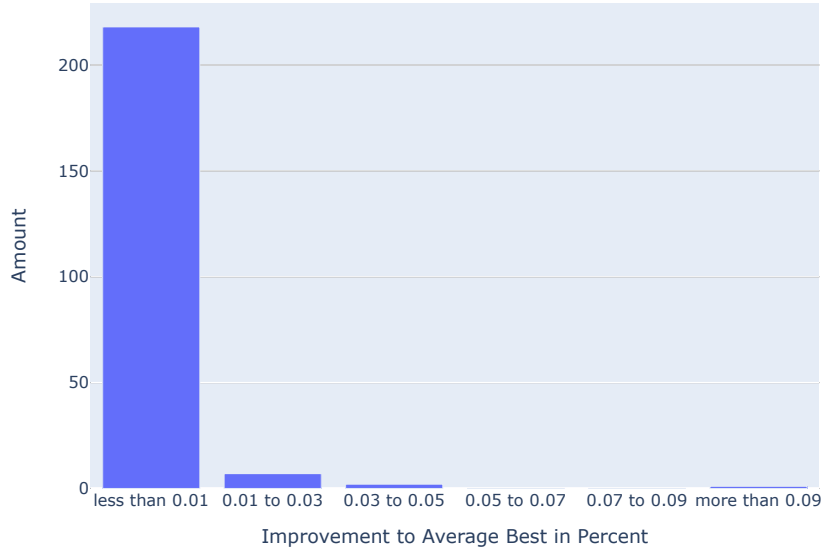

Figure C.5.: Improvement for Best Imputation Method per Experiment Scenario Relative to the Average Best Imputation Method by Percentage

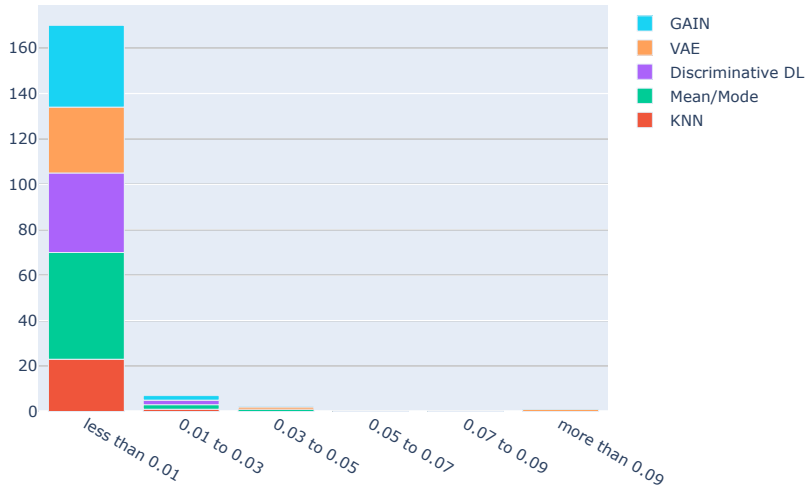

Figure C.6.: Improvement for Best Imputation Method per Experiment Scenario Relative to the Average Best Imputation Method by Percentage by Imputation Method

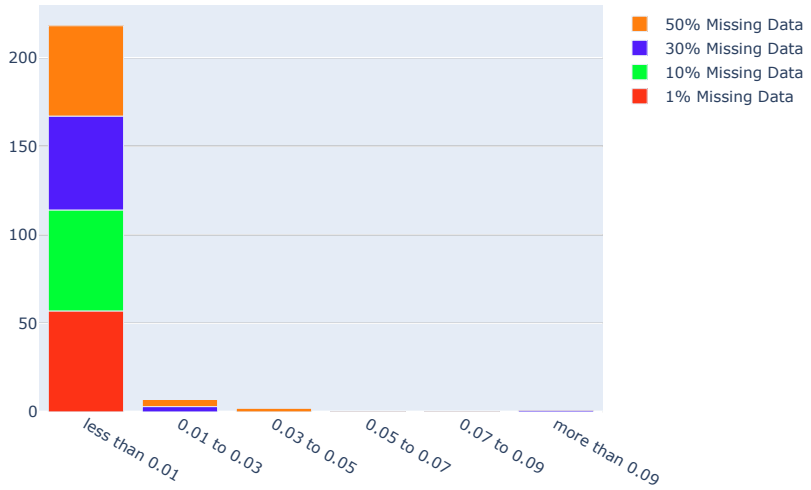

Figure C.7.: Improvement for Best Imputation Method per Experiment Scenario Relative to the Average Best Imputation Method by Percentage by Missing Fraction

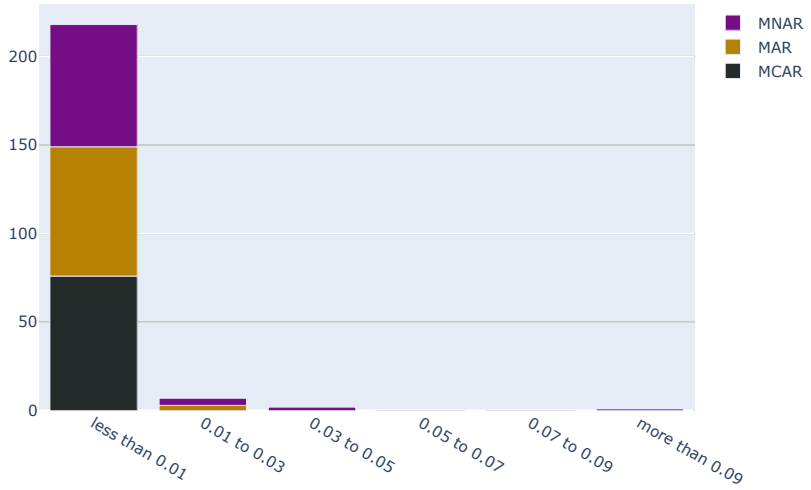

Figure C.8.: Improvement for Best Imputation Method per Experiment Scenario Relative to the Average Best Imputation Method by Percentage by Missingness Pattern

D. Appendix Binary Classification Subset Experiments

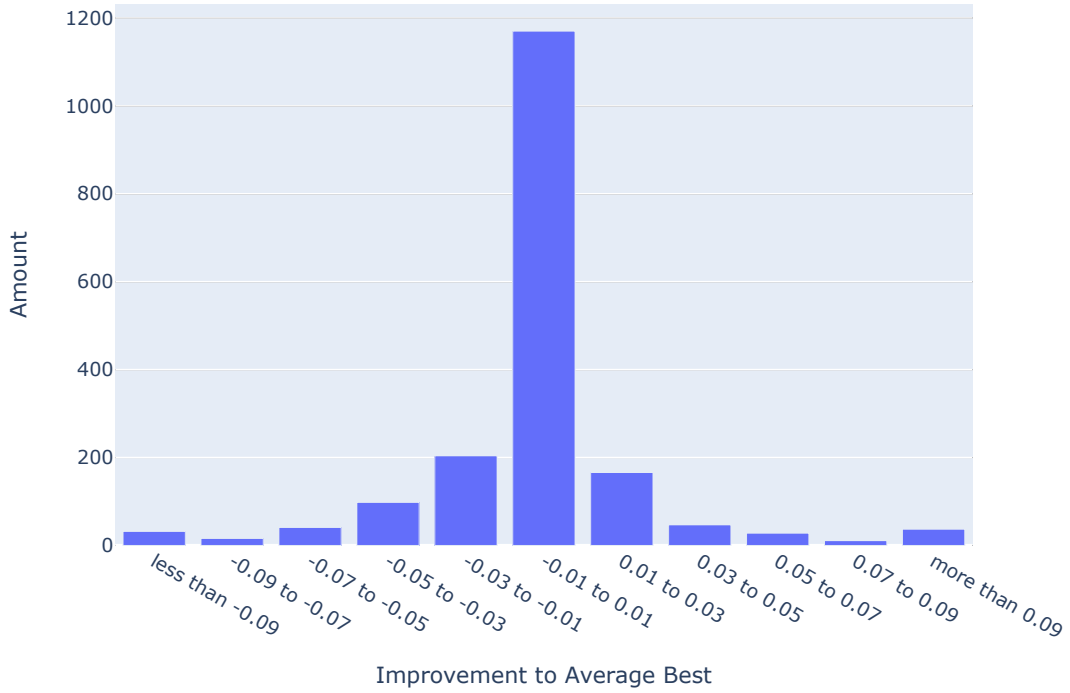

Figure D.1.: Improvement for Experiments Relative to the Average Best Imputation Method by F1 Score Points

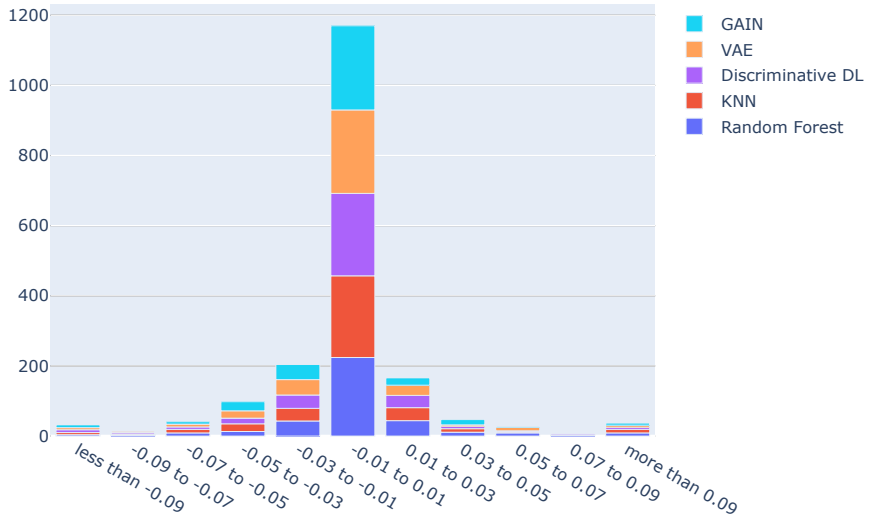

Figure D.2.: Improvement for Experiments Relative to the Average Best Imputation Method by F1 Score Points by Imputation Methods

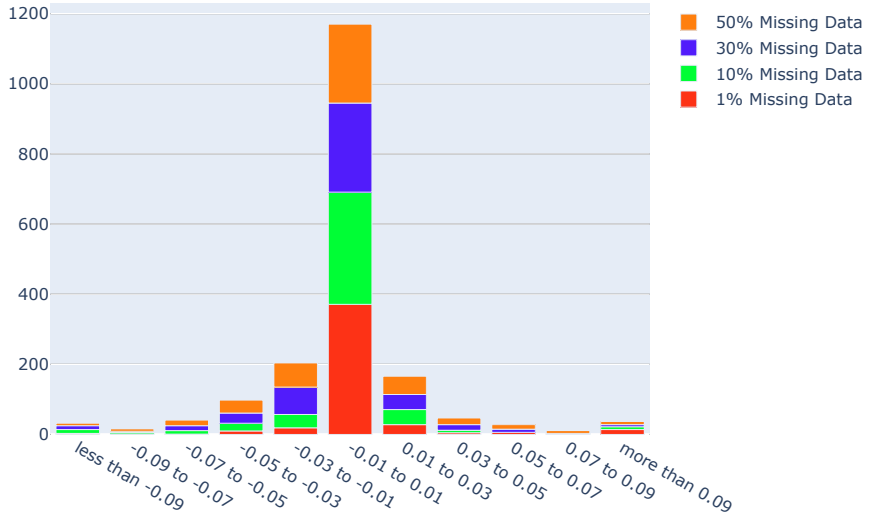

Figure D.3.: Improvement for Experiments Relative to the Average Best Imputation Method by F1 Score Points by Missing Fraction

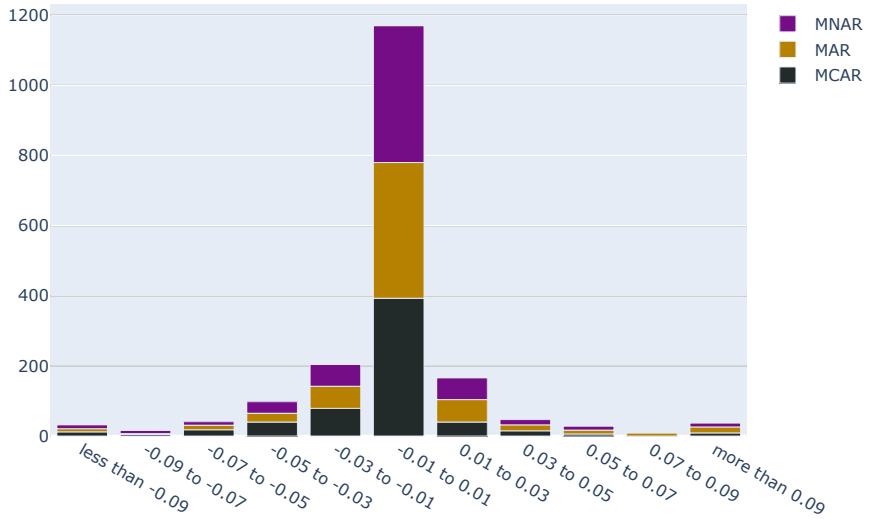

Figure D.4.: Improvement for Experiments Relative to the Average Best Imputation Method by F1 Score Points by Missingness Pattern

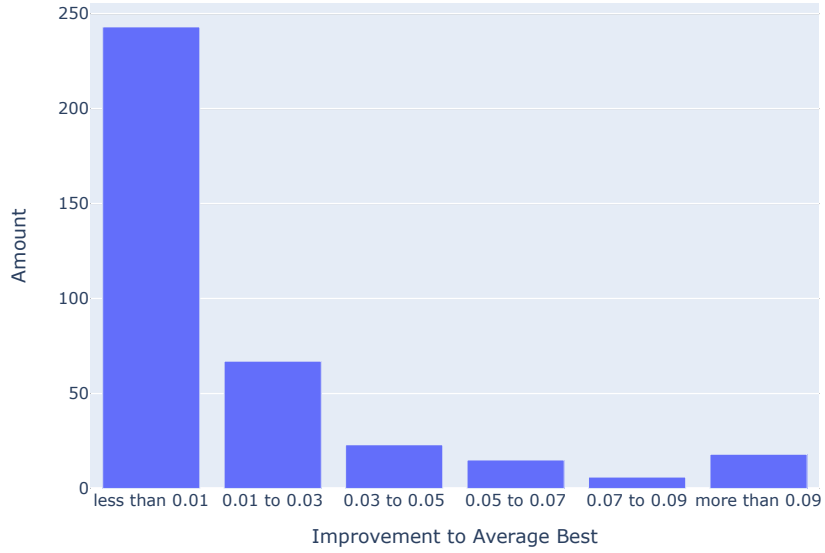

Figure D.5.: Improvement for Best Imputation Method per Experiment Scenario Relative to the Average Best Imputation Method by F1 Score Points

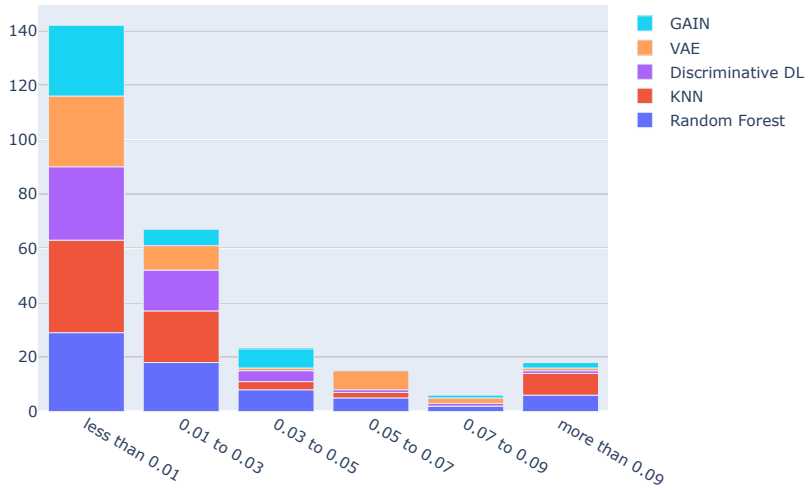

Figure D.6.: Improvement for Best Imputation Method per Experiment Scenario Relative to the Average Best Imputation Method by F1 Score Points by Imputation Method

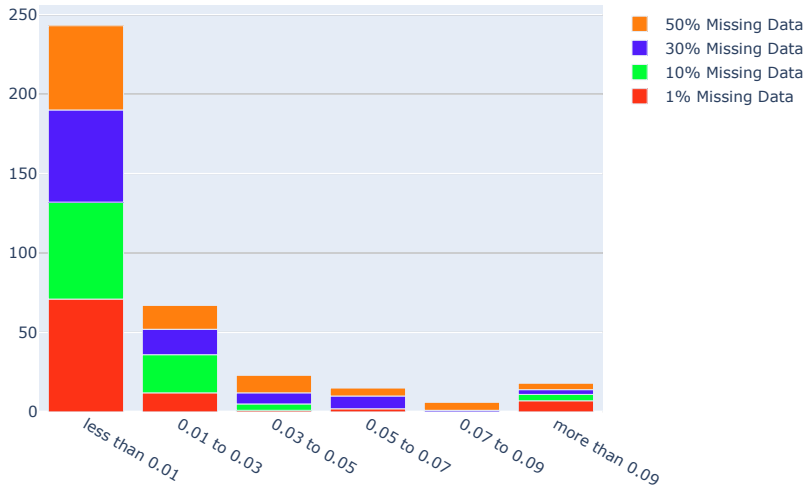

Figure D.7.: Improvement for Best Imputation Method per Experiment Scenario Relative to the Average Best Imputation Method by F1 Score Points by Missing Fraction

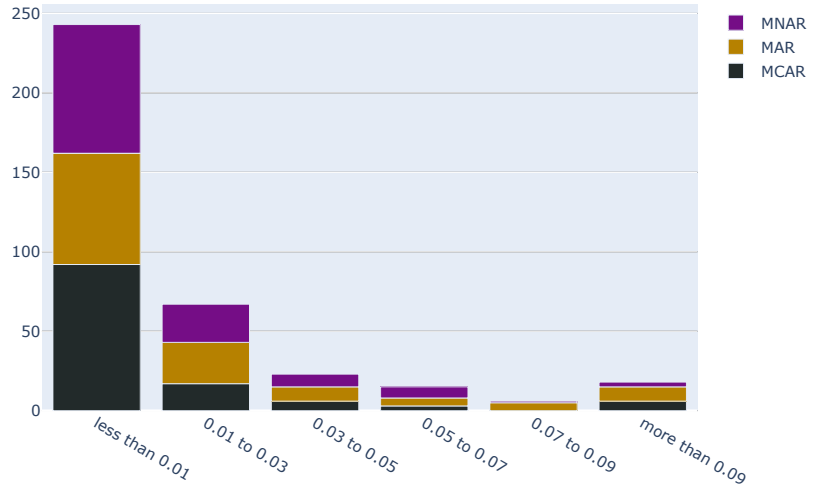

Figure D.8.: Improvement for Best Imputation Method per Experiment Scenario Relative to the Average Best Imputation Method by F1 Score Points Missingness Pattern

E. Appendix Multiclass Classification Subset Experiments

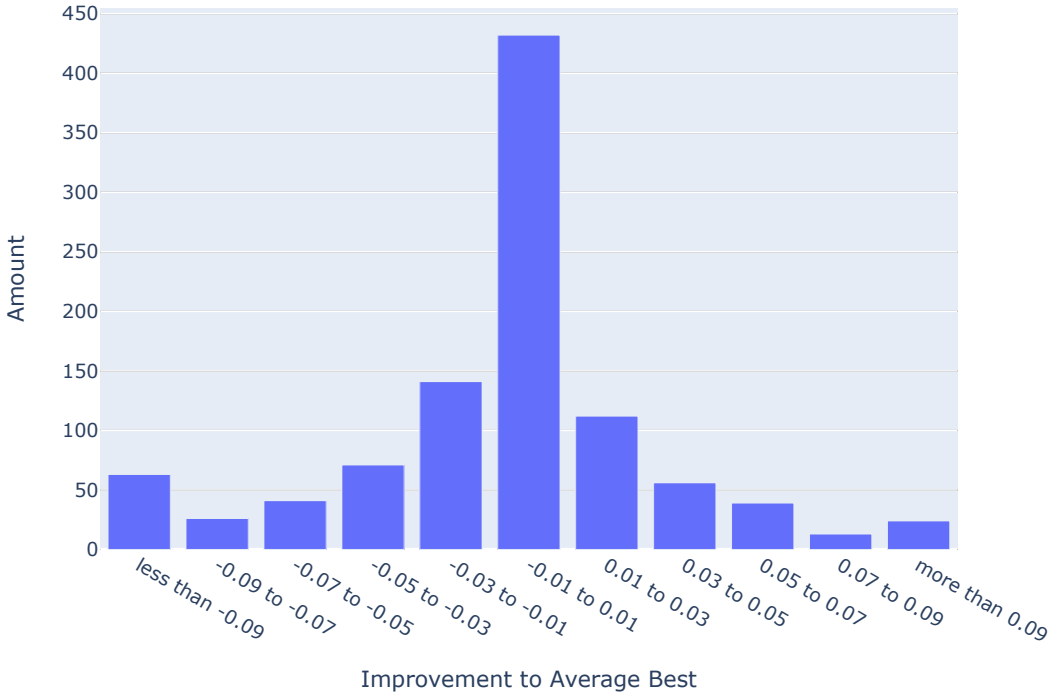

Figure E.1.: Improvement for Experiments Relative to the Average Best Imputation Method by F1 Score Points

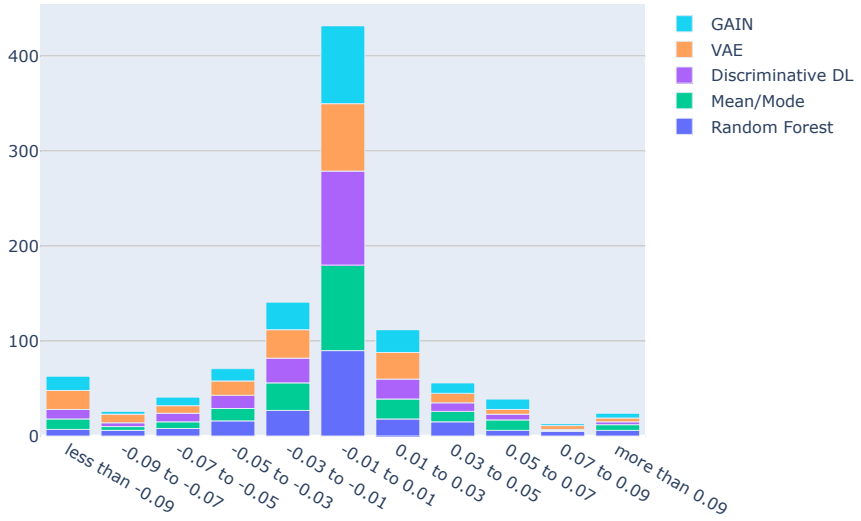

Figure E.2.: Improvement for Experiments Relative to the Average Best Imputation Method by F1 Score Points by Imputation Method

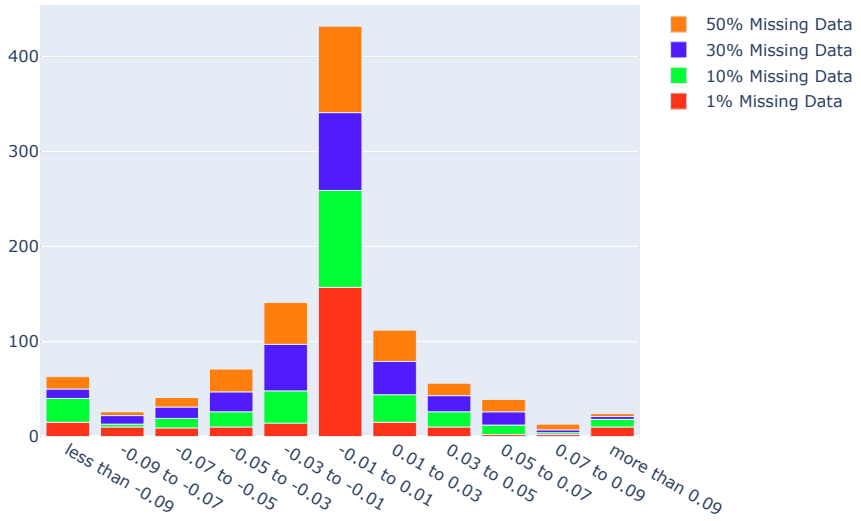

Figure E.3.: Improvement for Experiments Relative to the Average Best Imputation Method by F1 Score Points by Missing Fraction

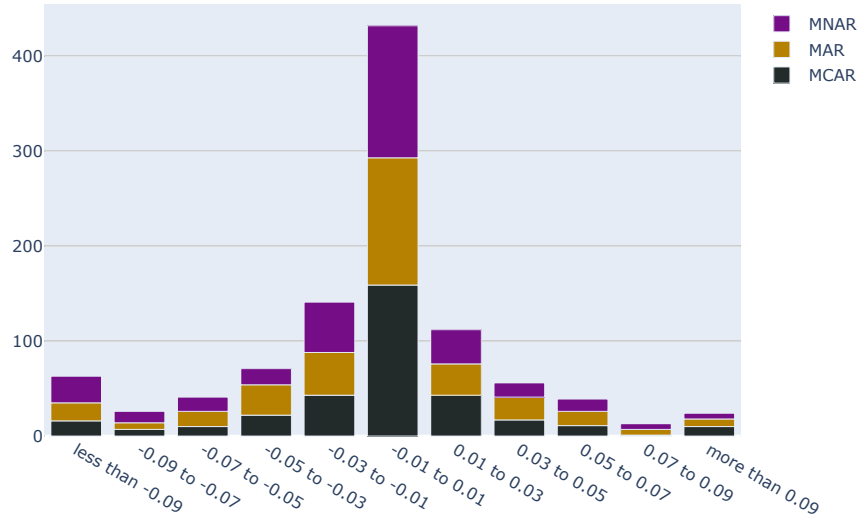

Figure E.4.: Improvement for Experiments Relative to the Average Best Imputation Method by F1 Score Points by Missingness Pattern

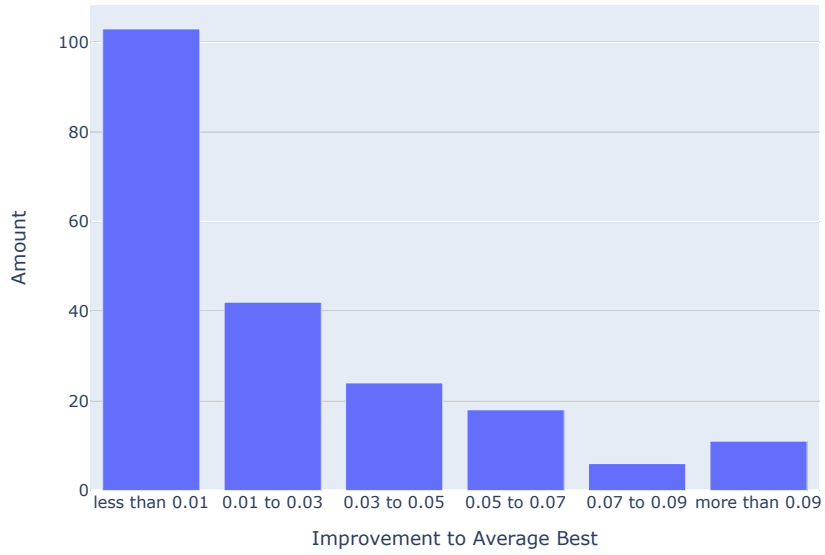

Figure E.5.: Improvement for Best Imputation Method per Experiment Scenario Relative to the Average Best Imputation Method by F1 Score Points

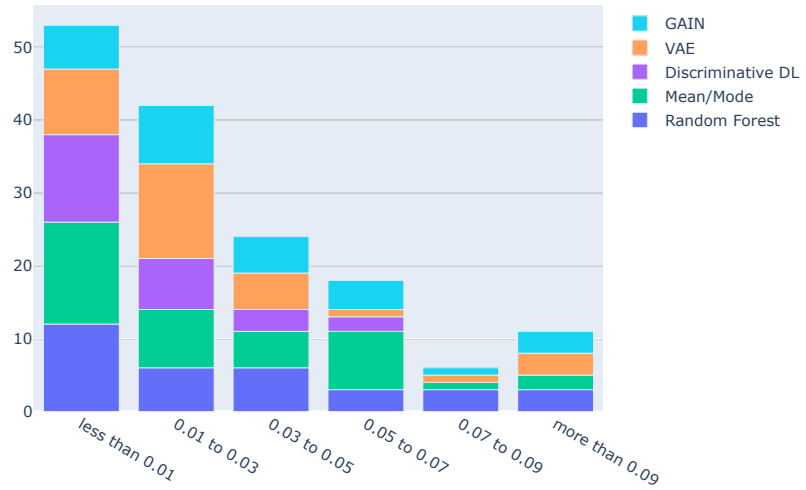

Figure E.6.: Improvement for Best Imputation Method per Experiment Scenario Relative to the Average Best Imputation Method by F1 Score Points by Imputation Method

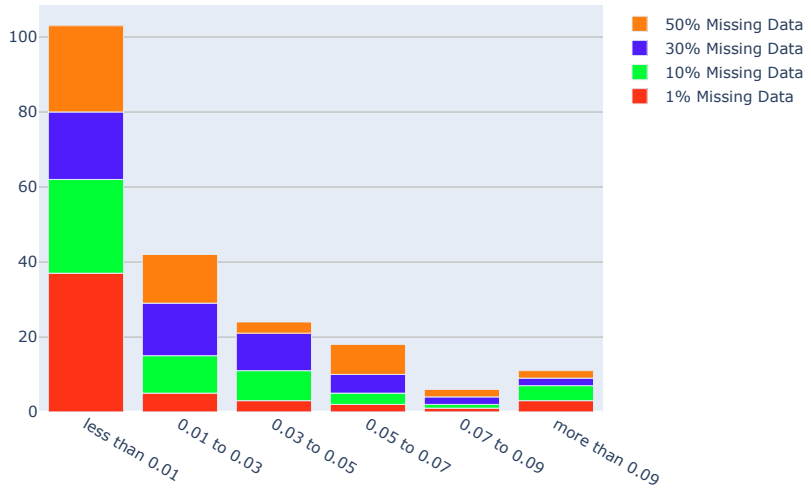

Figure E.7.: Improvement for Best Imputation Method per Experiment Scenario Relative to the Average Best Imputation Method by F1 Score Points by Missing Fraction

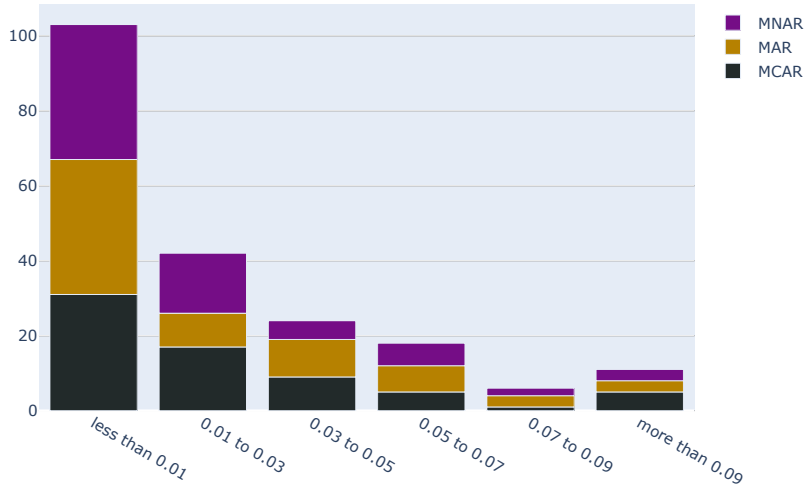

Figure E.8.: Improvement for Best Imputation Method per Experiment Scenario Relative to the Average Best Imputation Method by F1 Score Points by Missingness Pattern

F. Appendix Regression Subset Experiments

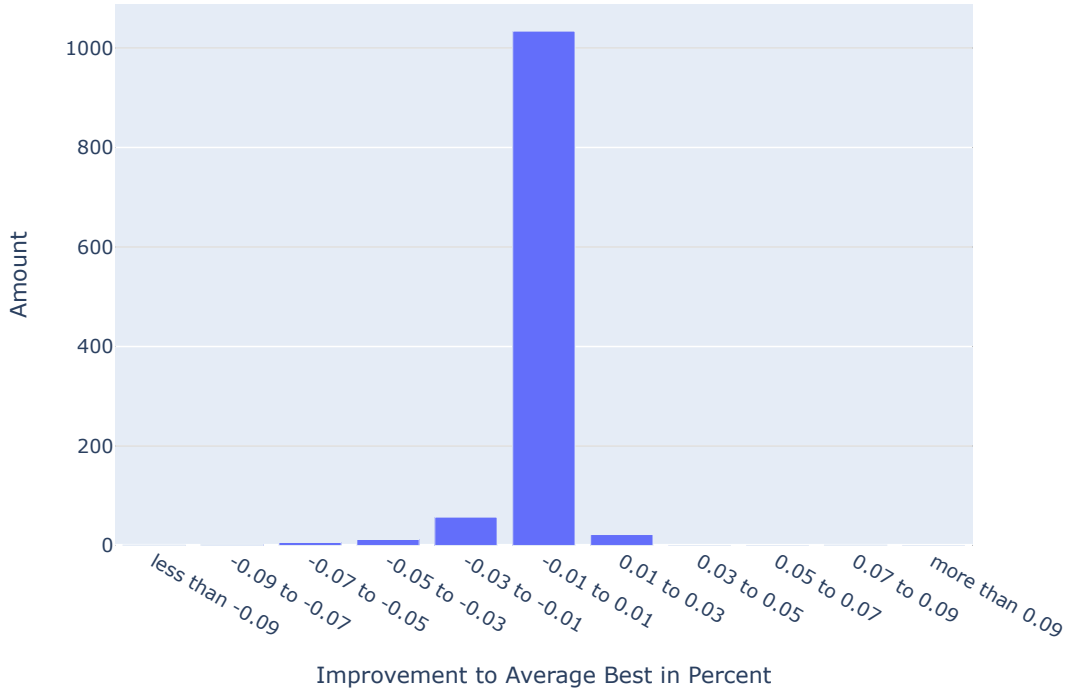

Figure F.1.: Improvement for Experiments Relative to the Average Best Imputation Method by Percentage

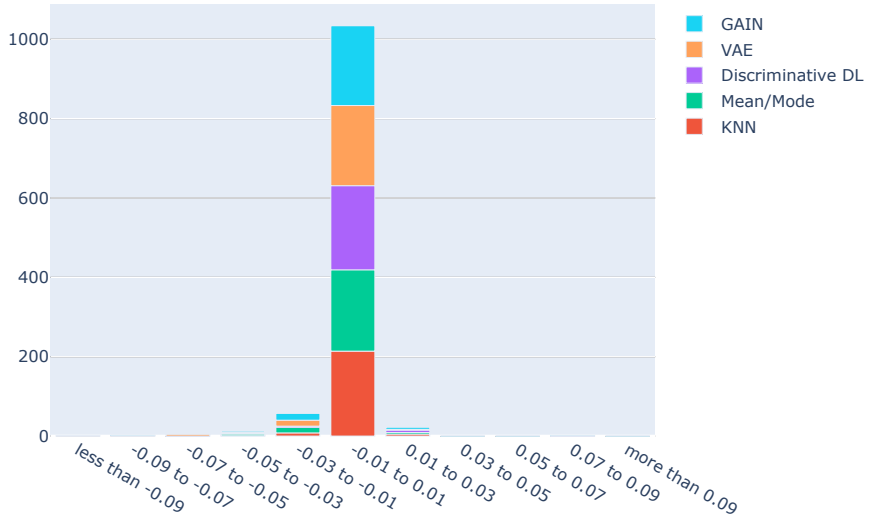

Figure F.2.: Improvement for Experiments Relative to the Average Best Imputation Method by Percentage by Imputation Method

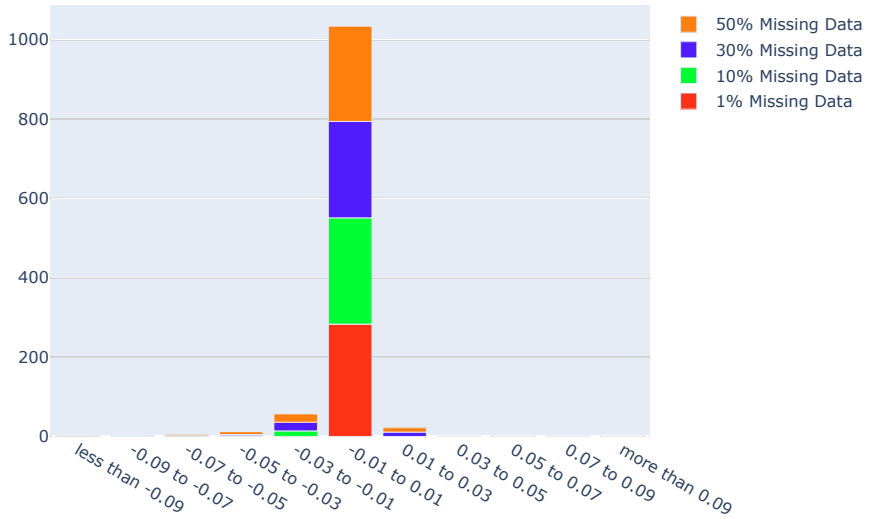

Figure F.3.: Improvement for Experiments Relative to the Average Best Imputation Method by Percentage by Missing Fraction

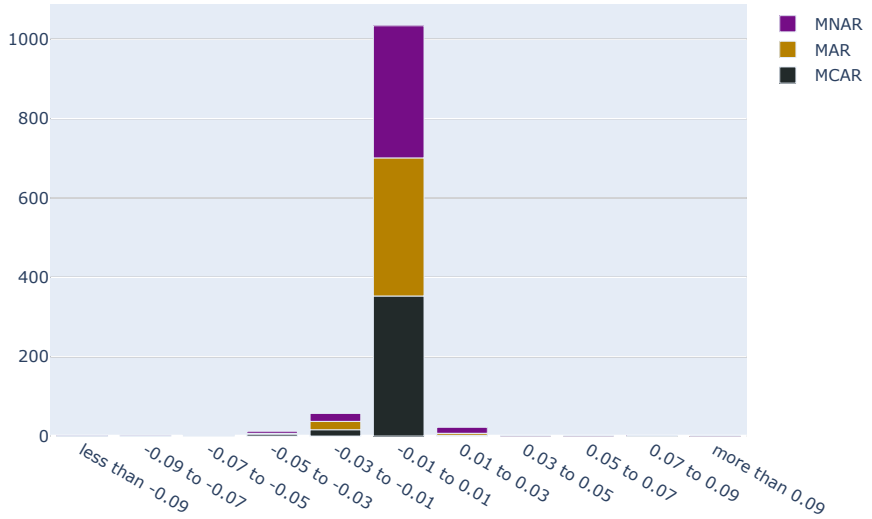

Figure F.4.: Improvement for Experiments Relative to the Average Best Imputation Method by Percentage by Missingness Pattern

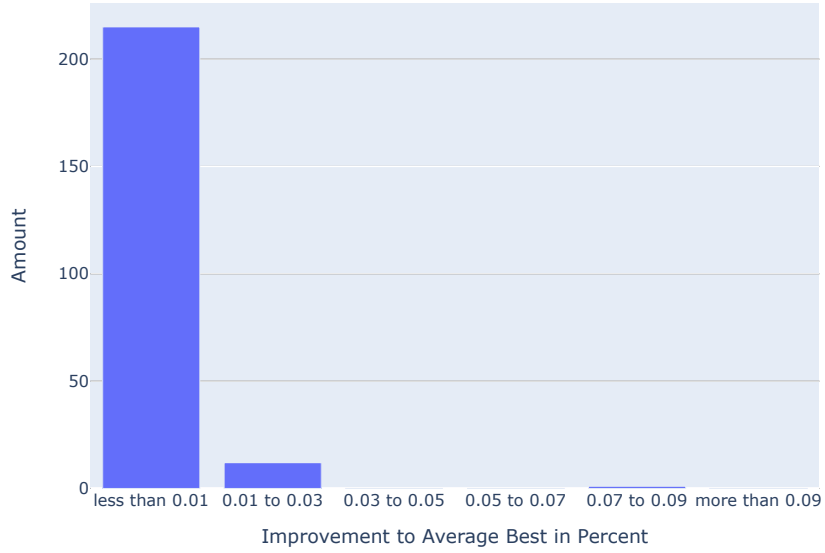

Figure F.5.: Improvement for Best Imputation Method per Experiment Scenario Relative to the Average Best Imputation Method by Percentage

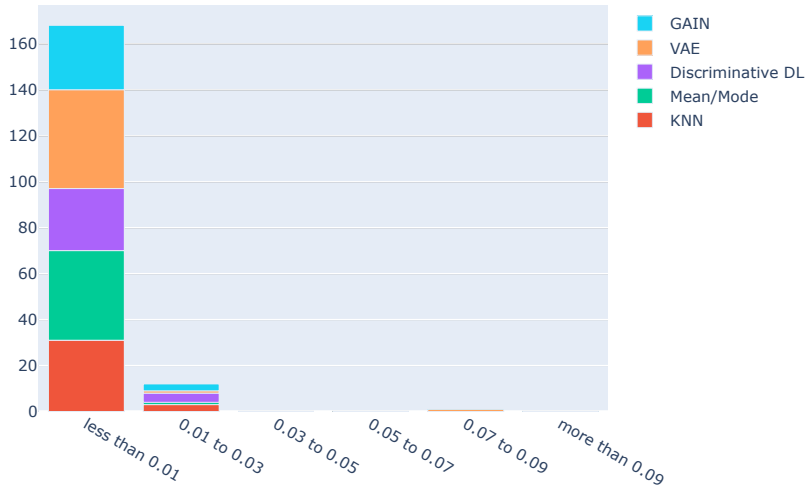

Figure F.6.: Improvement for Best Imputation Method per Experiment Scenario Relative to the Average Best Imputation Method by Percentage by Imputation Method

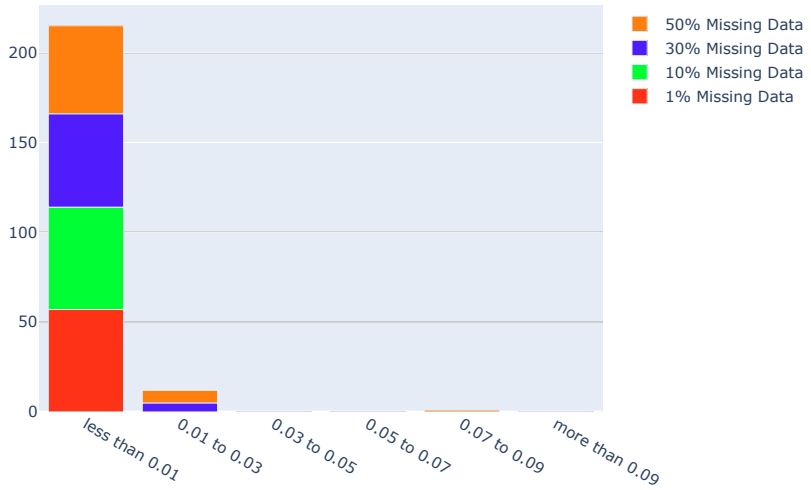

Figure F.7.: Improvement for Best Imputation Method per Experiment Scenario Relative to the Average Best Imputation Method by Percentage by Missing Fraction

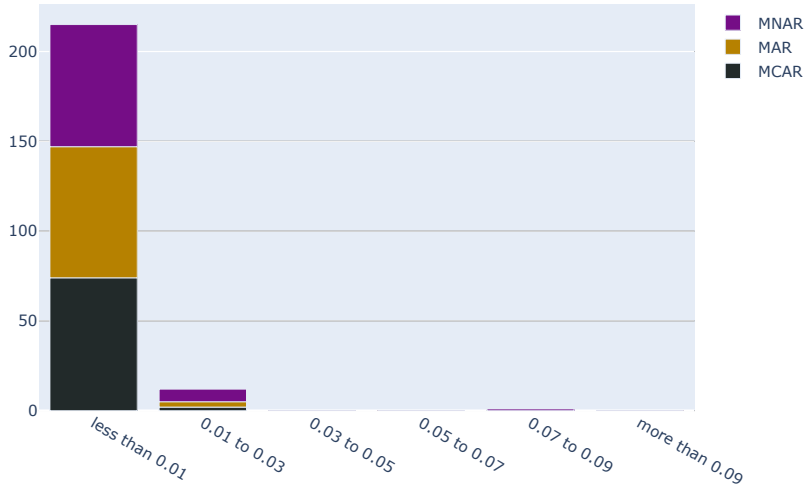

Figure F.8.: Improvement for Best Imputation Method per Experiment Scenario Relative to the Average Best Imputation Method by Percentage by Missingness Pattern

G. Appendix Binary Classification Baseline - Imputed Comparisons

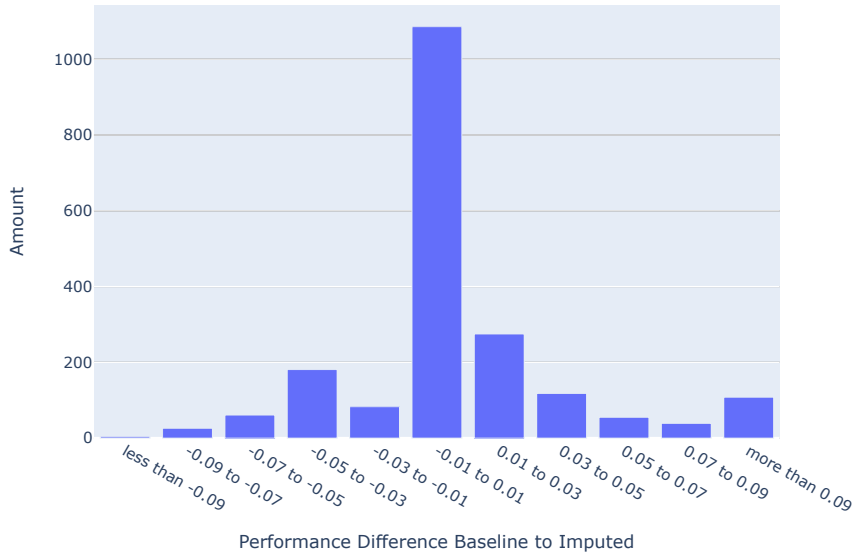

Figure G.1.: Improvements for the Baseline Model Relative to the Model Trained on Imputed Data

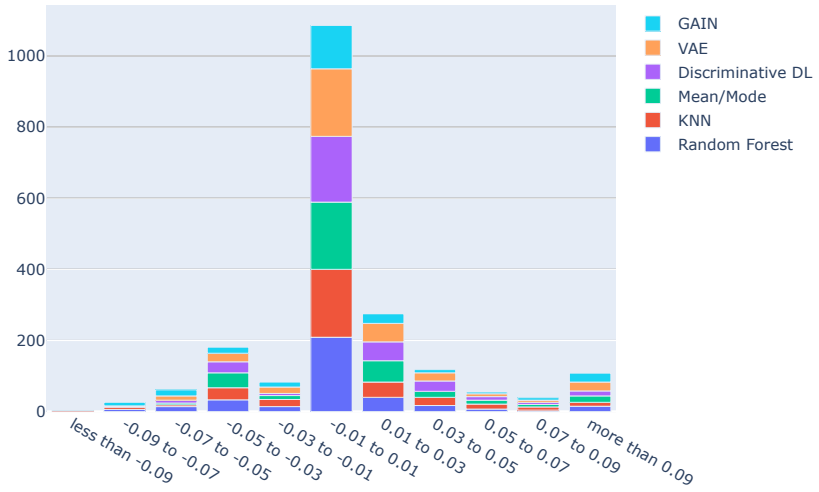

Figure G.2.: Improvements for the Baseline Model Relative to the Model Trained on Imputed Data by Imputation Method

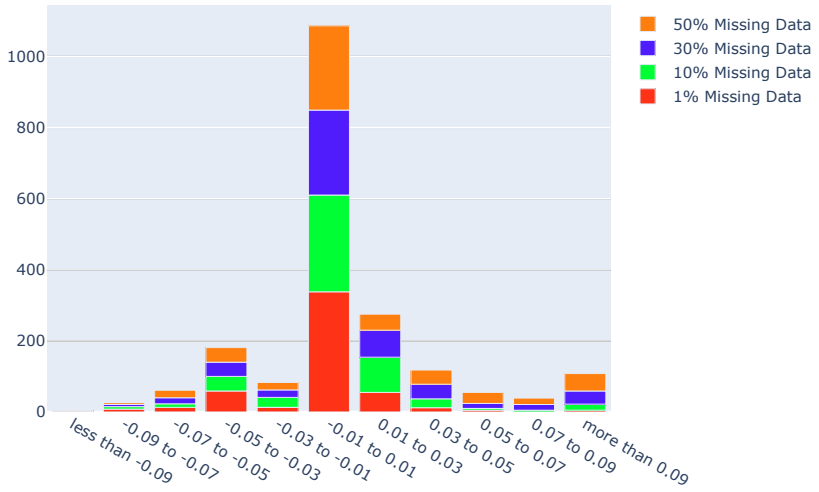

Figure G.3.: Improvements for the Baseline Model Relative to the Model Trained on Imputed Data by Missing Fraction

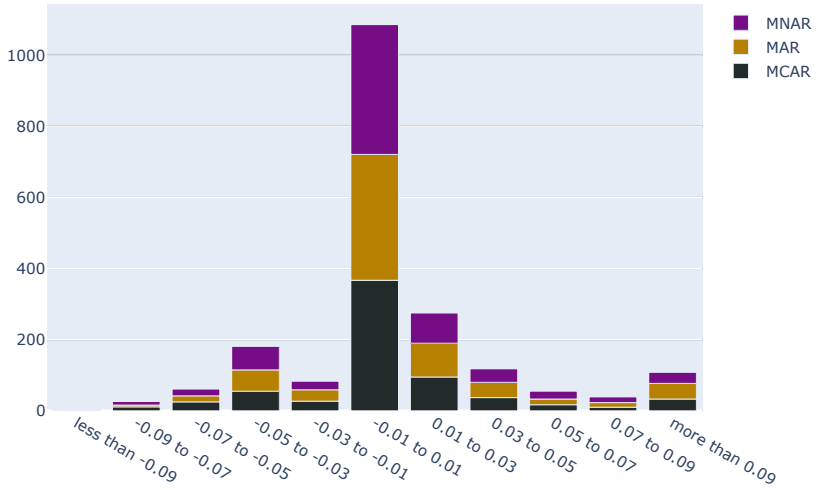

Figure G.4.: Improvements for the Baseline Model Relative to the Model Trained on Imputed Data by Missing Pattern

H. Appendix Binary Classification Imputed - Subset Comparisons

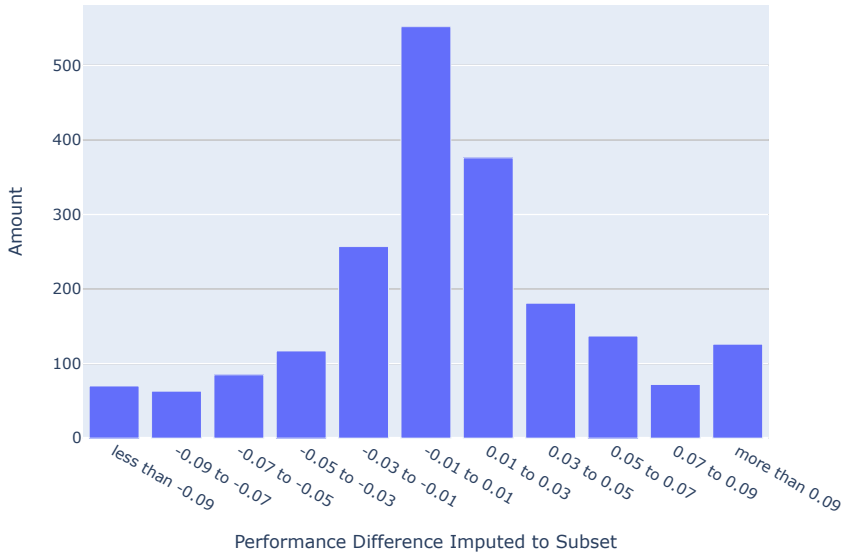

Figure H.1.: Improvements for the Model Trained on Imputed Data Relative to the Model Trained on Subset Data

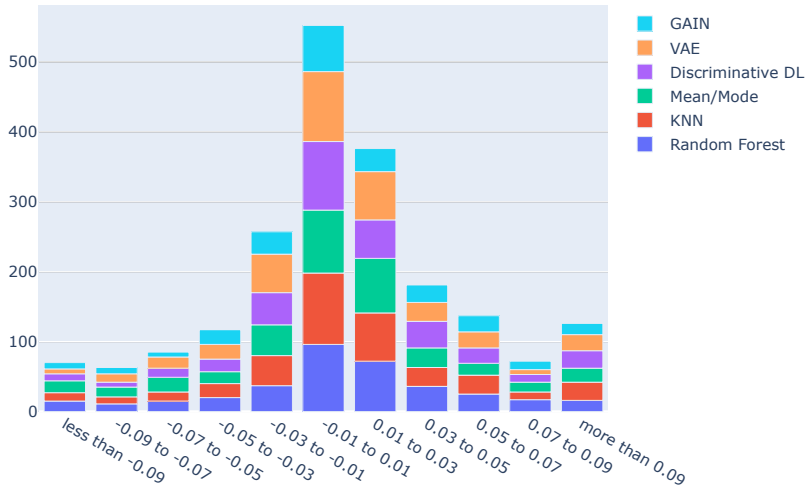

Figure H.2.: Improvements for the Model Trained on Imputed Data Relative to the Model Trained on Subset Data by Imputation Method

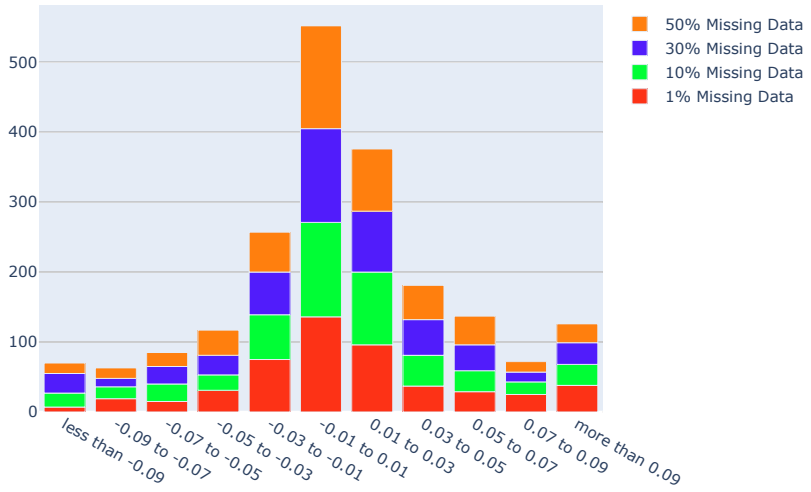

Figure H.3.: Improvements for the Model Trained on Imputed Data Relative to the Model Trained on Subset Data by Missing Fraction

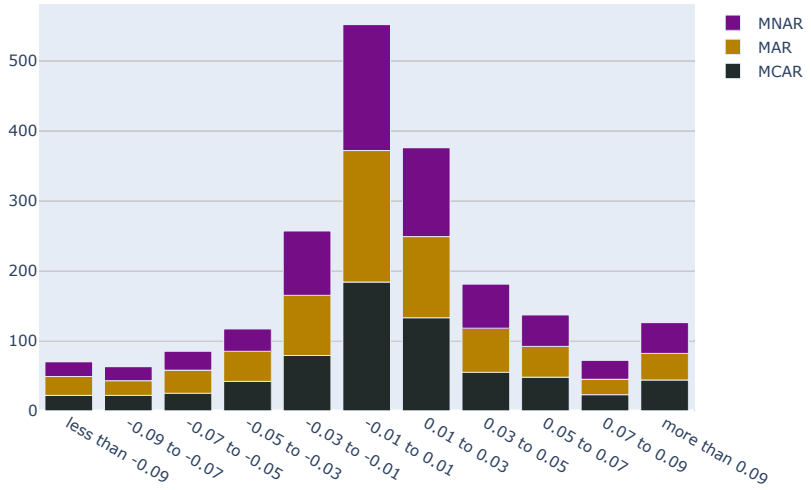

Figure H.4.: Improvements for the Model Trained on Imputed Data Relative to the Model Trained on Subset Data by Missing Pattern

I. Appendix Multiclass Classification Baseline - Imputed Comparisons

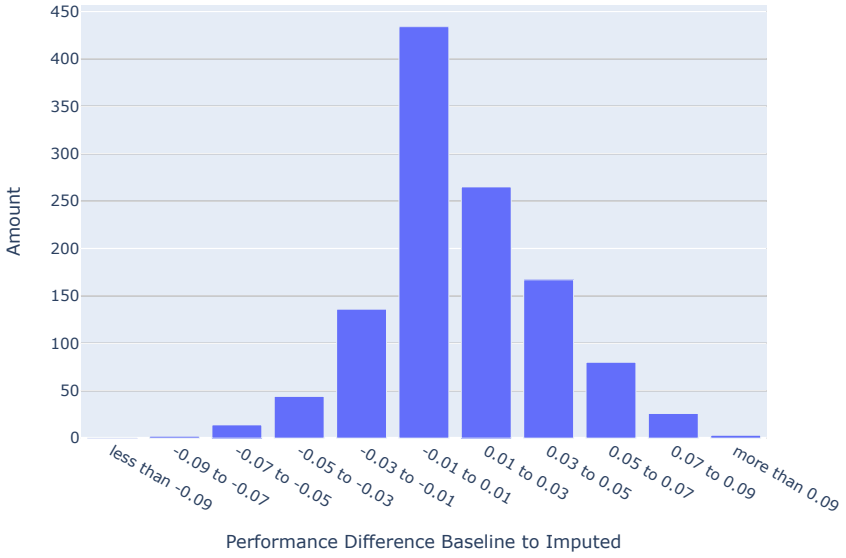

Figure I.1.: Improvements for the Baseline Model Relative to the Model Trained on Imputed Data

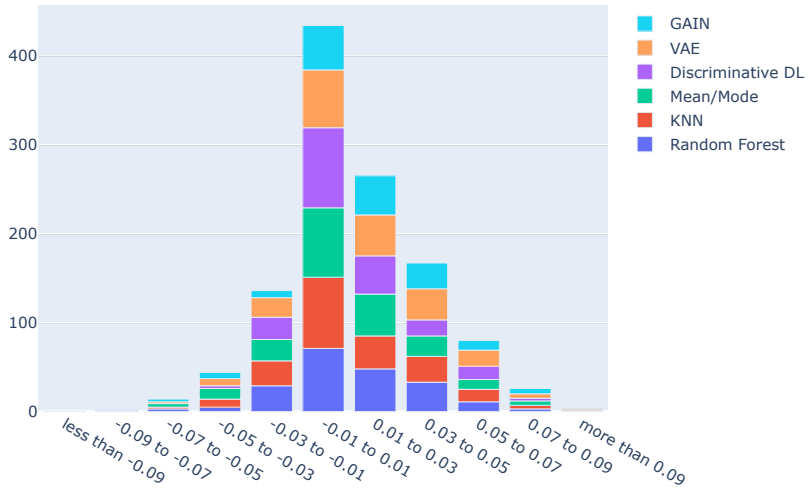

Figure I.2.: Improvements for the Baseline Model Relative to the Model Trained on Imputed Data by Imputation Method

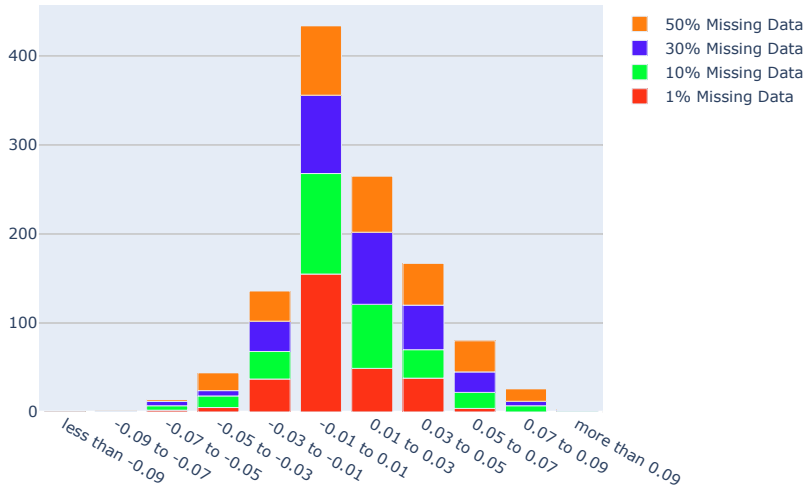

Figure I.3.: Improvements for the Baseline Model Relative to the Model Trained on Imputed Data by Missing Fraction

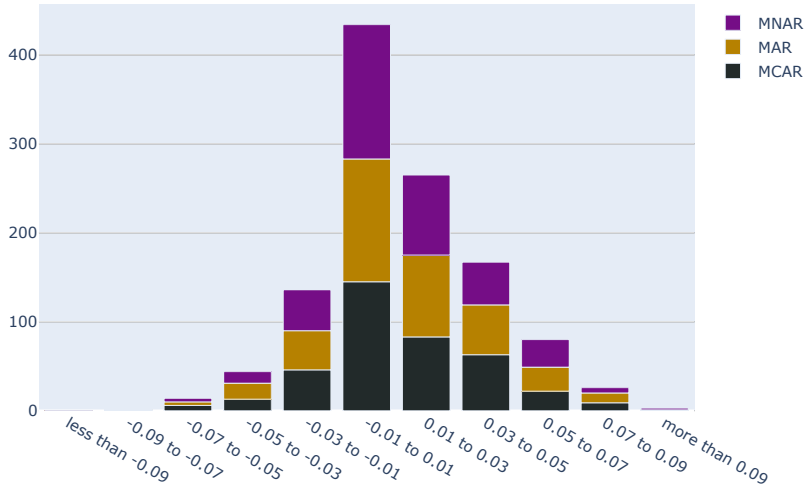

Figure I.4.: Improvements for the Baseline Model Relative to the Model Trained on Imputed Data by Missing Pattern

J. Appendix Multiclass Classification Imputed - Subset Comparisons

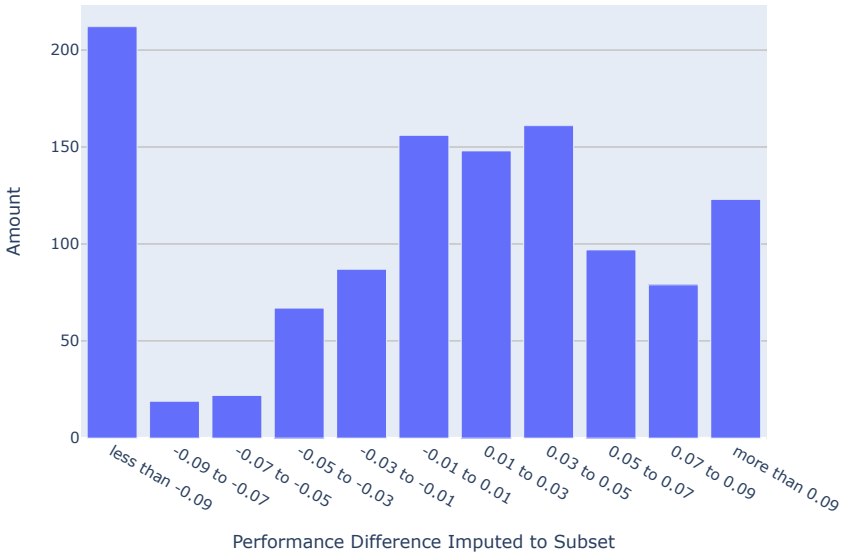

Figure J.1.: Improvements for the Model Trained on Imputed Data Relative to the Model Trained on Subset Data

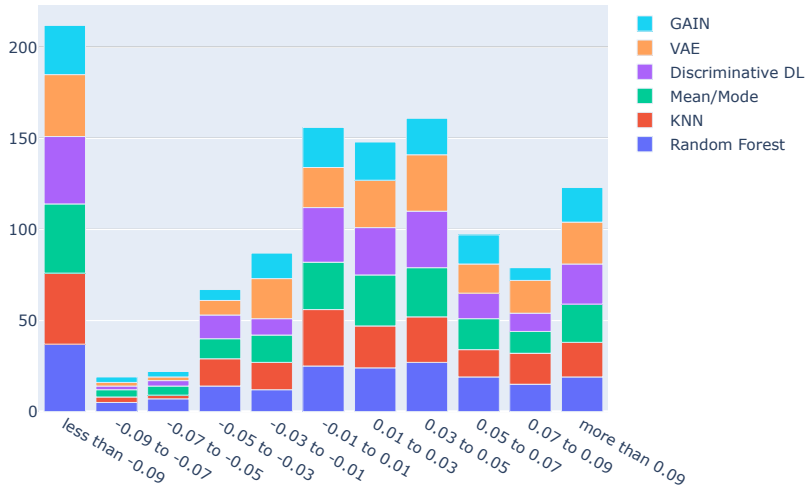

Figure J.2.: Improvements for the Model Trained on Imputed Data Relative to the Model Trained on Subset Data by Imputation Method

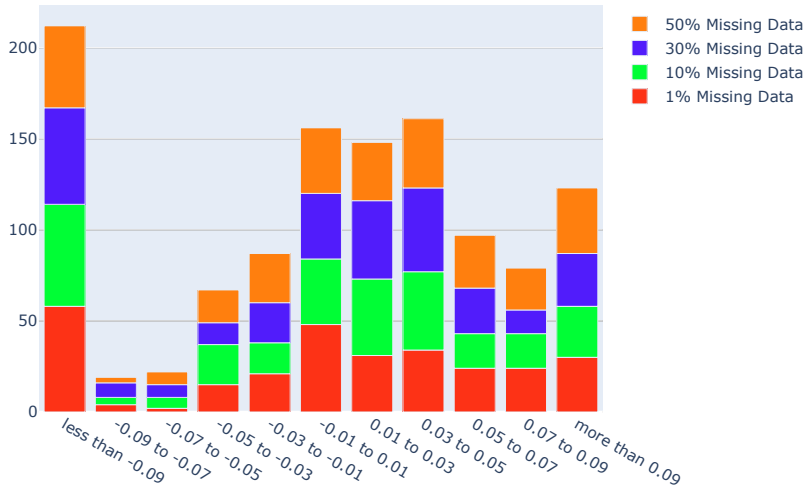

Figure J.3.: Improvements for the Model Trained on Imputed Data Relative to the Model Trained on Subset Data by Missing Fraction

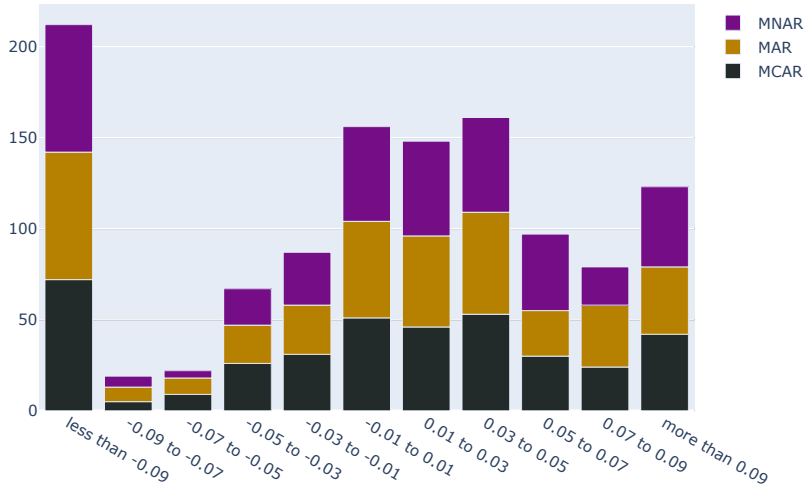

Figure J.4.: Improvements for the Model Trained on Imputed Data Relative to the Model Trained on Subset Data by Missing Pattern

K. Appendix Regression Baseline - Imputed Comparisons

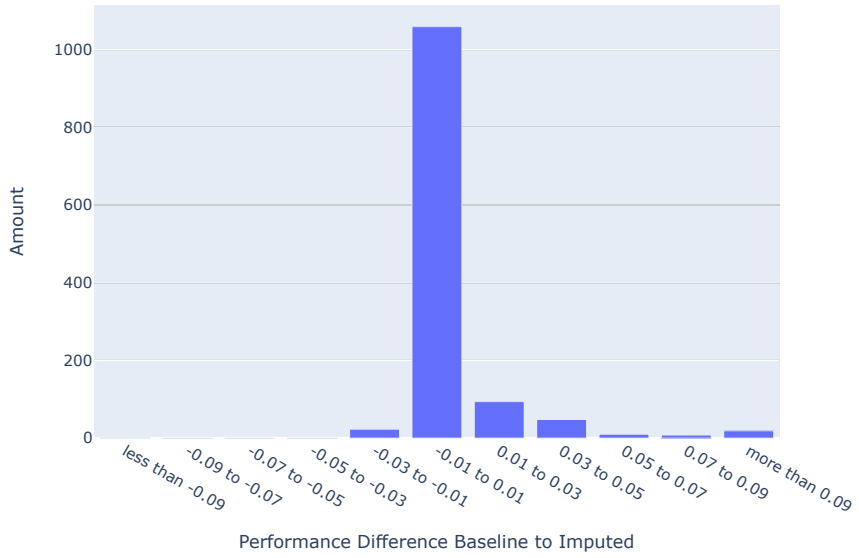

Figure K.1.: Improvements for the Baseline Model Relative to the Model Trained on Imputed Data

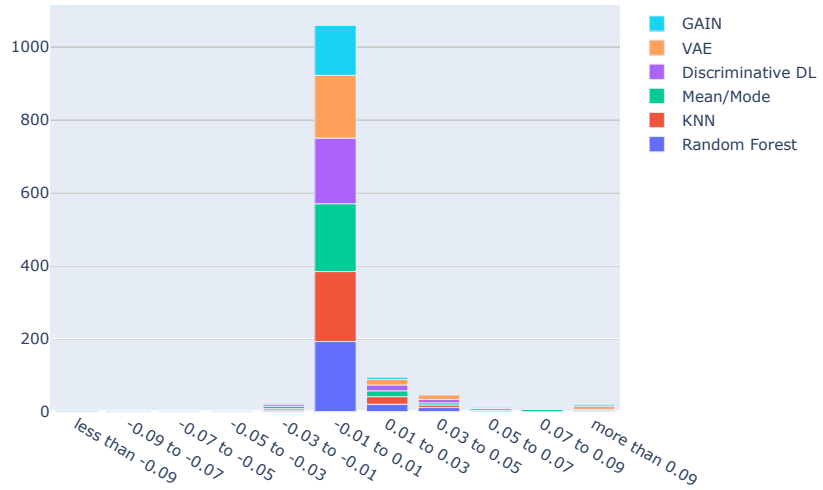

Figure K.2.: Improvements for the Baseline Model Relative to the Model Trained on Imputed Data by Imputation Method

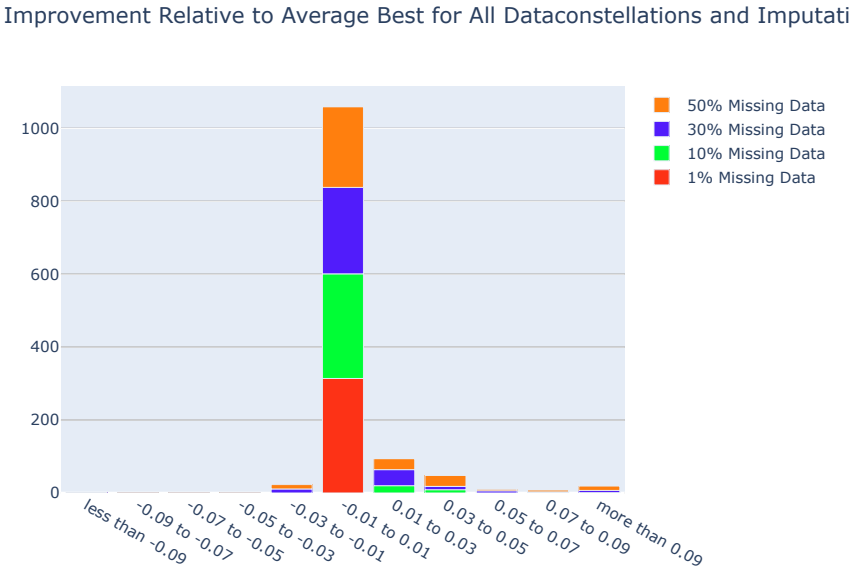

Figure K.3.: Improvements for the Baseline Model Relative to the Model Trained on Imputed Data by Missing Fraction

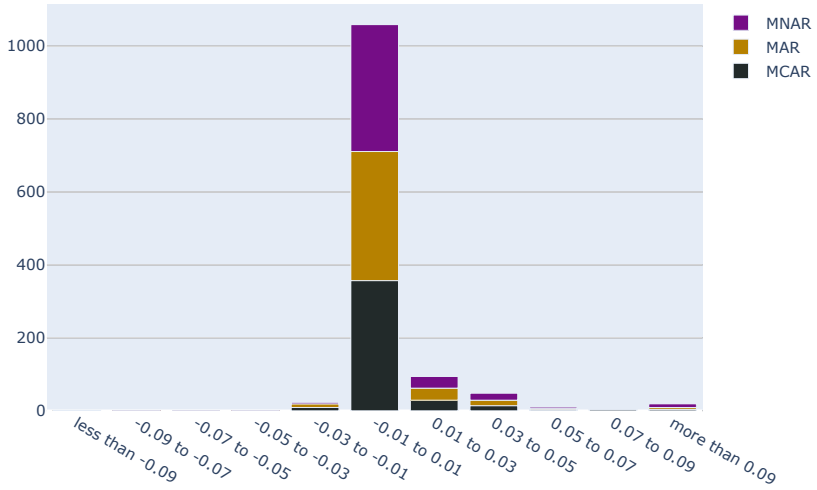

Figure K.4.: Improvements for the Baseline Model Relative to the Model Trained on Imputed Data by Missing Pattern

L. Appendix Regression Imputed - Subset Comparisons

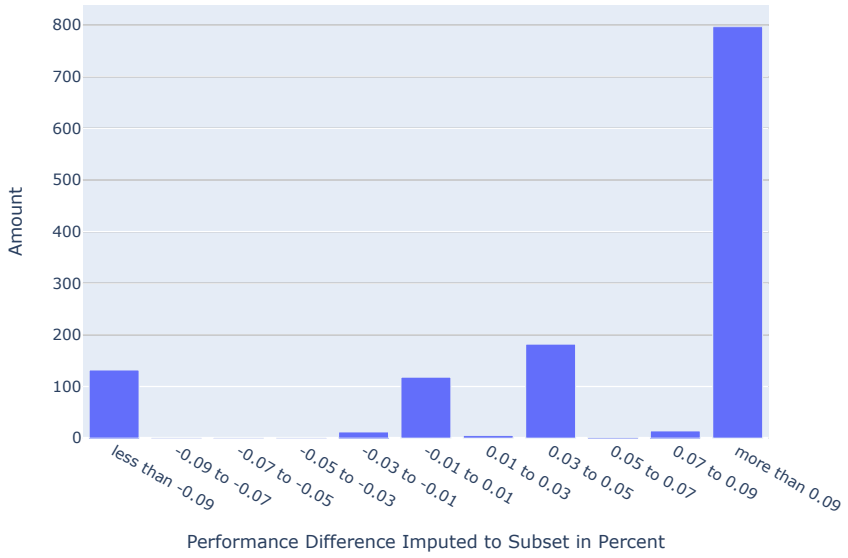

Figure L.1.: Improvements for the Model Trained on Imputed Data Relative to the Model Trained on Subset Data

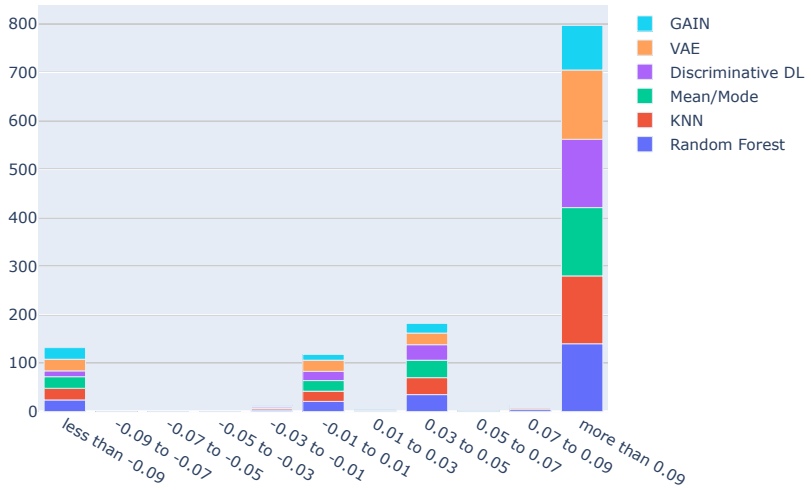

Figure L.2.: Improvements for the Model Trained on Imputed Data Relative to the Model Trained on Subset Data by Imputation Method

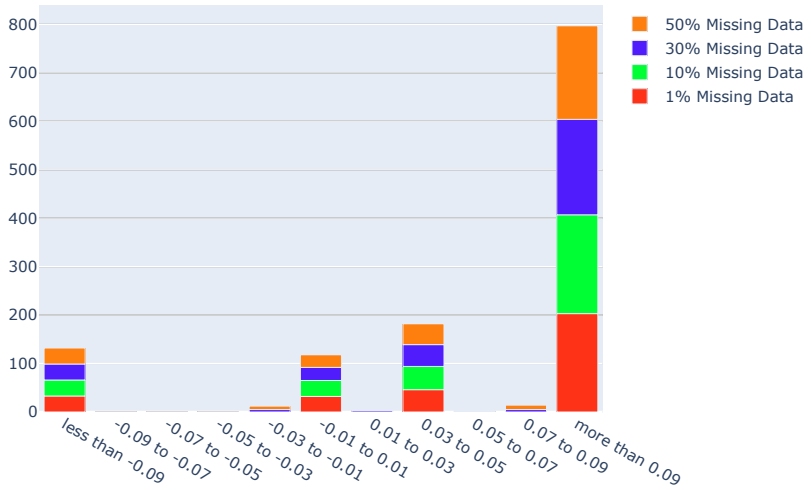

Figure L.3.: Improvements for the Model Trained on Imputed Data Relative to the Model Trained on Subset Data by Missing Fraction

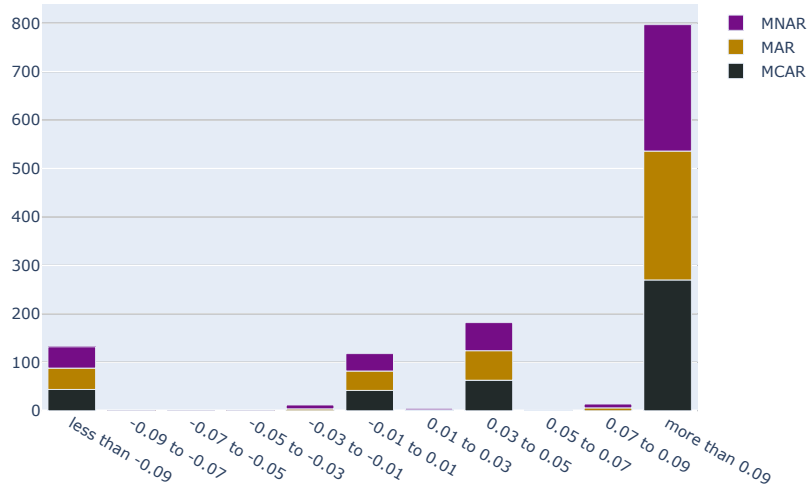

Figure L.4.: Improvements for the Model Trained on Imputed Data Relative to the Model Trained on Subset Data by Missing Pattern

M. Important Changes in the Framework

Change File Path: src/jenga/src/jenga/basis.py

Change Location: class Task / def fit_baseline_model

Preparation for changed model training (with imputed data) by skipping the previously implemented *SimpleImputer*.

```
        categorical_preprocessing = Pipeline(
            [
                #('mark_missing',
                   SimpleImputer(strategy='most_frequent')),
                ('one_hot_encode',
                   OneHotEncoder(handle_unknown='ignore'))
            ]
        )

        numerical_preprocessing = Pipeline(
            [
                #('mark_missing', SimpleImputer(strategy='mean')),
                ('scaling',  StandardScaler())
            ]
        )
```

Listing M.1: Preparation for Model Training with Imputed Data

Change File Path: src/jenga/src/jenga/basis.py

Change Location: BinaryClassificationTask & MultiClassClassification & RegressionTask / def _get_pipeline_grid_scorer_tuple

To ensure reproducibility the random seed is set for *SGDClassifier* and *SGDRegressor*.

```
        rng = np.random.RandomState(5)
        print(rng, "random seed for SGD Classifier")
        pipeline = Pipeline(
            [
                ('features', feature_transformation),
                ('learner', SGDClassifier(max_iter=1000,
                    n_jobs=-1, random_state=rng))
            ]
        )
```

Listing M.2: SGDClassifier Random Seed

```
        rng = np.random.RandomState(5)
        print(rng, "random seed for SGD #regressor")
        pipeline = Pipeline(
            [
                ('features', feature_transformation),
                ('learner', SGDRegressor(max_iter=1000,
                    random_state=rng))
            ]
        )
```

Listing M.3: SGDRegressor Random Seed

Change File Path: src/jenga/src/jenga/basis.py

Change Location: class TabularCorruption / def sample_rows

Changed the approach to how the dependent column for MAR is picked. The original code rearranges the items randomly every time, that is bypassing the static set random seed.

```
    elif self.sampling.endswith('AR'):
        columns_temp_ar = data.loc[:, data.columns != self.column]
        list_for_columns = list(columns_temp_ar.columns)
        depends_on_col = np.random.choice(list_for_columns)
        print(depends_on_col)
        rows =
            data[depends_on_col].sort_values().iloc[perc_idx].index
```

Listing M.4: Column Selection MAR Experiments

Change File Path: src/jenga/src/jenga/tasks/openml.py

Change Location: class OpenMLTask

Set static random seed for training/test split to ensure that experiments are deterministic.

```
class OpenMLTask(Task):

    def __init__(self, openml_id: int, train_size: float = 0.8,
        seed: Optional[int] = None):

        seed=32
        X, y = fetch_openml(data_id=openml_id, as_frame=True,
            return_X_y=True, cache=False)
        train_data, test_data, train_labels, test_labels =
            train_test_split(X, y, train_size=train_size)
        categorical_columns = [
            column for column in X.columns
            if pd.api.types.is_categorical_dtype(X[column])
        ]
        numerical_columns = [
            column for column in X.columns
            if pd.api.types.is_numeric_dtype(X[column]) and
                column not in categorical_columns
        ]


        super().__init__(
            train_data=train_data,
            train_labels=train_labels,
            test_data=test_data,
            test_labels=test_labels,
            categorical_columns=categorical_columns,
            numerical_columns=numerical_columns,
            is_image_data=False,
            seed=seed
        )
```

Listing M.5: OpenML Random Seed

Change File Path: src/jenga/src/jenga/tasks/openml.py

Change Location: class OpenMLTask

Adjustment for Subset Experiment, which selects randomly 10% of the dataset for further experiments. The random seed is set to ensure that the experiment remains deterministic.

```
class OpenMLTask(Task):

    def __init__(self, openml_id: int, train_size: float = 0.8,
        seed: Optional[int] = None):


        X, y = fetch_openml(data_id=openml_id, as_frame=True,
            return_X_y=True, cache=False)

        before_subset = pd.concat([X, y], axis=1)

        np.random.seed(32)
        len_dataset = len(before_subset)
        print(len_dataset,'Length of dataset')

        df_subset = before_subset.sample(frac=0.10)
        y_df = df_subset.iloc[:,-1:]
        y = y_df.iloc[:,0]
        X = df_subset[df_subset.columns[:-1]]

        seed=32
        train_data, test_data, train_labels, test_labels =
            train_test_split(X, y, train_size=train_size)
        categorical_columns = [
            column for column in X.columns
            if pd.api.types.is_categorical_dtype(X[column])
        ]
        numerical_columns = [
            column for column in X.columns
            if pd.api.types.is_numeric_dtype(X[column]) and
                column not in categorical_columns
        ]
```

Listing M.6: Subset Experiment Adjustments

Change File Path: src/data_imputation_paper/experiment.py

Change Location: class Experiment / def run

The random seed is set to make experiments and multiple runs deterministic.

```
for task_id, task_class in self._task_id_class_tuples:
    self._result[task_id] = {}

    task = task_class(openml_id=task_id)

    for missing_type in self._missing_types:
        self._result[task_id][missing_type] = {}

        for missing_fraction in self._missing_fractions:
            self._result[task_id][missing_type][missing_fraction]
                = {}

            for strategy in self._strategies:

                experiment_path = self._base_path / f"{task_id}"
                    / missing_type / f"{missing_fraction}" /
                    f"{strategy}"

                try:
                    evaluator =
                        self.strategy_to_EvaluatorClass[strategy](
                        task=task,
                        missing_fraction=missing_fraction,
                        missing_type=missing_type,
                        target_column=target_column,
                        imputer_class=self._imputer_class,
                        imputer_args=self._imputer_arguments,
                        path=experiment_path,
                        seed=42
                    )
                    evaluator.evaluate(self._num_repetitions)
                    result = evaluator._result
```

Listing M.7: Random Seed for Multiple Repetitions of Experiments

Change File Path: src/data_imputation_paper/evaluation_corrupted.py

Change Location: Class Evaluator / def evaluate

Adjust function so that now the imputed training and test data is used, instead of corrupted training and test data.

```
1
2  train_data_corrupted, test_data_corrupted = self._discard_values(
3      task=self._task,
4      to_discard_columns=self._discard_in_columns,
5      missing_fraction=self._missing_fraction,
6      missing_type=self._missing_type,
7  )
8
9  imputer = self._imputer_class(**self._imputer_arguments)
10
11 start_time = time.time()
12 imputer.fit(train_data_corrupted.copy(), [target_column])
13 elapsed_time = time.time() - start_time
14
15 train_imputed, train_imputed_mask =
       imputer.transform(train_data_corrupted)
16 test_imputed, test_imputed_mask =
       imputer.transform(test_data_corrupted)
17
18 if result_temp._baseline_performance is None:
19     base_model =
           self._task.fit_baseline_model(train_imputed.copy(),
           self._task.train_labels)
20     self._task._baseline_model = base_model
21
22     predictions = self._task._baseline_model.predict(test_imputed)
23     result_temp._baseline_performance =
           self._task.score_on_test_data(predictions)
24
25 # NOTE: masks are DataFrames => append expects Series
26 result_temp.append(
27     target_column=target_column,
28     train_data_imputed=train_imputed,
29     test_data_imputed=test_imputed,
30     test_data_corrupted=test_data_corrupted,
31     train_imputed_mask=train_imputed_mask[target_column],
32     test_imputed_mask=test_imputed_mask[target_column],
33     elapsed_time=elapsed_time,
34     best_hyperparameters=imputer.get_best_hyperparameters()
35 )
```

Listing M.8: Adjustment for the Usage of the Imputed Training and Test Data