# Imputing missing values for mixed-type tabular data sets using generative adversarial networks

Applied Data Science Masters Thesis

**Ouassim Bannany**
o.bannany@students.uu.nl

**First supervisor**
Dr. A.A.A. (Hakim) Qahtan

**Second supervisor**
Dr. M.W. (Mel) Chekol

**Universiteit Utrecht**

Department of Information and Computing Sciences
Utrecht University
Netherlands
July 2022

# Abstract

Missing values are common in real-world data sets and represent a challenging problem in performing most data analytics tasks. For that reason, many data imputation techniques have been proposed in the past to fill the missing values. However, these existing techniques may not capture the characteristics of the data and mislead the data analytics techniques, resulting in inaccurate conclusions. Generative Adversarial Networks (GANs) proved to be a good technique for generating synthetic data; using GANs, synthetic examples are generated that preserve the existing values in the record. Then, these synthetic examples can be utilized to fill the missing values and capture the data characteristics better than the other data imputation techniques. This project implements a framework based on Generative Adversarial Networks to impute the missing values for a given incomplete data set. The performance of the framework is evaluated using two different methodologies: 1) By determining the prediction error of the imputed values after introducing missing values in an otherwise complete data set, and 2) by comparing the performance of a classifier trained on a post-imputed data set, which has been imputed using our proposed framework and other imputation frameworks. The proposed framework outperformed other state-of-the-art frameworks at higher missing rates of 50% and beyond while achieving comparable results at lower missing rates. In addition, classifiers built using this proposed framework may lead to higher accuracy- and ROC-AUC scores compared with some of the other baseline methods.

***Keywords***— GAN, MAR, MCAR, MNAR

# Acknowledgements

This section is dedicated to those who aided me during my master thesis. First and foremost, I would like to express my sincere gratitude to the first supervisor Dr. A.A.A. (Hakim) Qahtan, for always being available for questions, even outside of regular working hours. In addition, I would like to thank him for his excellent guidance, open discussions, and for aiding me in overcoming obstacles that I faced during the thesis period. I would like to extend my gratitude to the second supervisor Dr. M.W. (Mel) Chekol, for his time, feedback, and interest in my thesis project. Next, I would like to thank my fellow students for their participation in the open discussions during project meetings and their valuable insights on this topic. Lastly, words can not express the endless gratitude that I owe to my parents and family. Their motivation and support throughout my academic career are the reason for my academic accomplishments.

# Contents

# Chapter 1

# Introduction

The primary goal of data analytics is to uncover patterns and extract valuable insights from raw data. These insights can then be used to improve decision-making, optimize- and automatize processes, and more. However, the quality of the raw data is of great importance in the data analytics pipeline since the trait of poor quality data will inevitably carry over through this pipeline process and lead to a degradation of the results[11]. "Garbage in, garbage out" is the common phrase used by computer scientists to emphasize that the output quality of a program strictly relies on the input quality. Therefore, researchers should aim to raise the quality of their utilized data to get more sensible end results. One critical factor that can provide an indication on the quality of the utilized data is the completeness of the data set. Depending on the missing data mechanism, incomplete data can result in wrong conclusions being drawn due to biased inferences and reduce the quality of models that have been trained using this data[7][8]. In addition, many predictive models such as neural networks and support vector machines require complete data during the training phase, in which case the researcher is required to handle the missing values beforehand. Since the worldwide data volume is predicted to grow substantially in the upcoming years[15], it can be expected that this problem will grow alongside the increasing data supply. Hence, it is essential that better imputation techniques are developed, which can impute the missing values more accurately in the raw data sets while retaining their statistical properties. Ultimately, this could raise the quality of the input data, prevent wrong conclusions from being drawn, increase the quality of the models post-imputation, and generally lead to better results and insights to be obtained by the researcher.

In detail, missing values can be categorized as either explicit- or implicit missing values. In the case of explicit missing values, the absence is known and marked in the data set, e.g., 'NA' or 'NULL'. Whereas in the case of implicit missing values, the absence is unknown and disguised under a fake value in the data set. For instance, if a person does not want to disclose his/her salary in a survey but can not leave the question blank, '0' could be a typical fake value to hide the actual salary. Nevertheless, frameworks such as FAHES[10] can be utilized to detect these explicit missing values and transform them into implicit ones. Before dealing with the missing data problem, it is essential first to determine the missingness mechanism behind these missing values since certain imputation techniques might only guarantee unbiasedness under specific assumptions on the missingness mechanism[12]. Typically, missing data can be categorized into three different types depending on the connection between missing- and observed values:

1. There is Missing Completely at Random (MCAR) where the values are missing for reasons unrelated to the observed variables. Thus, for all cases, the probability of having missing values is constant.
2. Secondly, there is Missing at Random (MAR), where values are missing for reasons that are related to the observed variables in the data. Hence, the probability of having missing values depends on the observed variables and may not be constant for all cases.
3. Lastly, there is Missing Not at Random (MNAR), where values are missing for reasons related to both unobserved and observed variables. Hence, this implies that the probability of missing values also depends on unobserved variables.

Once the missingness mechanism has been determined, an appropriate method can be chosen to handle the missing values. Of which some shall be mentioned next.

## 1.1 Motivation

Traditional techniques that handle missing data are often flawed, distort the statistical properties of the actual underlying data distribution, introduce bias into the data set, or simply do not provide the most optimal imputations. Straightforward widely-used techniques include methods such as listwise deletion, mean/mode imputation, and regression. However, each of these techniques suffers from the problems mentioned above. There are also more sophisticated and well-established imputation techniques, such as Multiple Imputation by Chained Equations (MICE)[3], and ones that rely on machine learning, such as missForest[14]. Although these techniques provide higher quality imputations that can better mimic the statistical properties of the observed data, more advanced methodologies for data imputations have been developed in recent years that rely on deep learning. Since deep learning techniques have shown great potential in other fields, this thesis aims to create a framework that similarly uses a deep learning approach for missing data imputation of mixed-type tabular data sets. Ultimately, the proposed framework could provide more accurate imputations in comparison with existing imputation frameworks and could thus be used to raise the data quality for researchers even further. Specifically, the proposed framework utilizes generative adversarial networks to solve the missing data problem more efficiently, which is one of those deep learning techniques. The GAN framework was initially introduced in 2014 by Goodfellow *et al.*[6]. This architecture consists of two neural networks, a generator and a discriminator, that contest against each other since each tries to maximize the opponent's loss. The generator is tasked with generating synthetic data that incorporates the characteristics of the training data. On the other hand, the discriminator is given a set of real and fake data and evaluates the probability of authenticity of this input data. After each training iteration, these neural networks indirectly learn from their opponent and aim to be more effective at their tasks. The end goal of the generator is to create synthetic data that the discriminator can not distinguish from the real data. In contrast, the discriminator tries to classify as accurately as possible whether the input data is real or fake. Assuming that the training phase has been successful, the generator should be able to create diverse, high-quality synthetic data, which the discriminator often fails to detect as fake data. Thus, any missing value in the row can be imputed by generating a complete synthetic row conditional on the observed variables. This ultimately results in a complete data set, which can be used for further analysis and predictive modeling.

## 1.2 Contributions

This thesis aims to provide valuable insights into the potential of using generative adversarial networks to solve the missing data problem in mixed-type tabular data sets. The following contributions to the field of missing data imputation are made: Firstly, a simple-to-use framework that utilizes GANs for data imputation has been built for researchers. The proposed framework has been automatized in several regards such that only the raw data is necessary as input. As a result, this framework could be more accessible to a larger audience of data specialists since it does not require researchers to have extensive knowledge of deep learning in order to be able to use the framework. The second contribution to the field is to provide insights into the effect of utilizing multiple GAN models on the reliability and quality of the imputations. In this proposed framework, three underlying GAN models each provide a candidate imputation for a missing value, and a final imputation is chosen based on a majority vote. Although this could provide more reliable imputations, the consideration needs to be made on whether the performance gain justifies the additional computational time. Lastly, the performance of the proposed framework is compared to various state-of-the-art imputation frameworks, using two different evaluation metrics. A novel evaluation metric is introduced where continuous- and categorical variables equally contribute to the overall prediction error score. The second evaluation metric determines the performance difference of a machine learning classifier post-imputation using the proposed framework in comparison with the other imputation frameworks.

# Chapter 2

# Related Work

In this chapter, relevant past research on missing data imputation has been dissected in more detail. Specifically, traditional methods and frameworks for dealing with missing values are discussed, together with possible shortcomings. Afterward, past research that utilized GANs in fields other than missing data imputation is mentioned, in addition to situations where GANs have failed during their training phases. Lastly, existing GAN frameworks for imputing missing data are mentioned and compared.

## 2.1 Traditional imputation techniques

In some cases, missing values can be deduced from other observed variables, such as a person's age if the date of birth is known. In cases where this is impossible, a commonly-used simplistic solution is to perform listwise deletion to eliminate missing values in the data set. In listwise deletion, any row that contains at least one missing value is discarded from the data set, which results in a smaller but complete data set. However, discarding all incomplete rows could result in biased results if the missingness mechanism is not MCAR. In addition, valuable information is wasted. If data is scarce, this could not be a feasible solution to the missing data problem as too little data might be left over for analysis. Therefore, a more viable solution to this problem is to impute the missing values with a guessed value. Data imputation techniques can typically be categorized as single- or multiple imputation. Single imputation does not account for the uncertainty of the predicted value since its goal is only to generate one guessed value per missing value. Well-known examples of single imputation techniques are mean/mode imputation and regression imputation. For mean imputation, the missing values are imputed with the mean value of the corresponding variable based on the observed values. Although simple to perform, this can severely underestimate the variance, especially when the number of missing values in a given variable is significant.

Additionally, utilizing mean imputation can result in biased estimates even when the missingness mechanism is MCAR. For regression imputation, on the other hand, a regression model is fit on the fully observed rows after listwise deletion and utilized to create a prediction for the missing values. Even though this may result in unbiased estimates under the MAR assumption, a disadvantage is that correlations are artificially increased, which consequently leads to the uncertainty of the predicted values being underestimated. Although these single imputation techniques are generally less complex, they each have drawbacks such as those mentioned above, making them impracticable for most use-cases. On the other hand, multiple imputation methods are more effective in imputing missing values since they account

for the uncertainty of the guessed values. Multiple complete data sets are generated from an incomplete data set, with each having (slightly) different imputations for the missing values. After analyzing these completed data sets, the results can be pooled to get one final estimated value to impute in the place of the missing value. In more recent years, machine learning approaches such as MissForest[14] have emerged with the same goal of imputing missing values. The framework works by first imputing the missing values with the variable's mean. Then a random forest model fits on each variable that initially contained missing values. The model then trains iteratively by considering the predicted values from the previous training iteration to improve the data quality. The MissForest framework has been shown to significantly outperform widely used imputation techniques by researchers, such as multiple imputation by chained equations (MICE)[3]. Nevertheless, more recently, deep learning techniques have shown even greater potential for effectively imputing missing data more accurately. Hence, the proposed framework of this study shall be compared to existing traditional frameworks such as MissForest and MICE.

## 2.2 Application of GANs in other fields

GANs have made compelling advancements in computer vision tasks since their introduction in 2014. This section discusses some of these frameworks, including their applications in society. Firstly, there is RenderGAN[13], which can generate labeled realistic images from unlabeled data. This framework allows researchers to perform supervised learning tasks on generated labeled images without manually labeling the data themselves or paying for data labeling services. Secondly, the cycleGAN[19] framework has been shown to be effective for image-to-image translation. It has been used for various creative purposes, such as photo enhancements and animal/object transfiguration. Nevertheless, GANs also form the foundation of controversial applications such as deepfake imagery and videos. Often, these deepfakes are used to spread fake news by generating synthetic videos of politicians and celebrities that are hardly distinguishable from reality. Although GANs have been successful in computer vision tasks, common problematic situations can occur when training the GAN architecture, which may ultimately lead to failure. Firstly, GANs can suffer from mode collapse, which is the case when the generator only learns to produce identical output rather than having variety in its generated data that mimics the training data distribution. Secondly, GANs often fail to converge during the training phase. One situation is when the discriminator becomes too strong to the point that it easily differentiates real data from generated data and thus dominates over the generator. This situation is problematic since it can result in the generator being unable to learn any further and generate more authentic data compared with earlier iterations, which is also known as the vanishing gradient problem[1]. On the other hand, the quality of the synthetic data might be low in the case where the discriminator is too weak to the point that the generator dominates. Hence, it is of great importance that the generator and discriminator converge to an equilibrium rather than dominating each other during training.

Lastly, we discuss the CTGAN[16] framework in more detail, which, similarly to this thesis, focuses on tabular data with a mix of continuous- and discrete variables but for a different purpose than missing data imputation. Instead, CTGAN is used for data augmentation tasks for situations where class imbalances are present in the data set. By generating synthetic data conditioned on the minority class, the class imbalance can be mitigated, which results in a data set that is more suitable as training data for statistical- and machine learning models. The authors elaborate on the difficulties of using traditional GAN frameworks to create synthetic data on mixed-type tabular data and propose solutions to these problems. Firstly, the authors propose an alternative approach to normalize continuous variables in the data set. Namely, a mode-specific normalization technique is introduced in place of traditional min-max normalization to transform continuous variables. The authors elaborate that continuous

variables often have complex multi-model non-Gaussian distributions and that min-max normalization of these complex distributions can result in a vanishing gradient, which can be evaded by utilizing this novel normalization technique. The second issue is that training the generator using highly imbalanced discrete variables often leads to mode collapse since using random sampling during training will not adequately represent rows of the minority class. For this reason, a more efficient sampling technique during the training phase is implemented to sample all the distinct values of categorical variables evenly. The authors first generate synthetic data using CTGAN and then use various existing GAN models to compare the quality of the synthetically generated data. Afterward, machine learning classifiers are trained using the synthetic training data, and the accuracy/F1 scores are compared. Classifiers trained on the CTGAN synthetic data achieved higher performance than other GAN models; hence the conclusion is made that the quality of the synthetic data of CTGAN is higher than these other frameworks. Although CTGAN has been designed for data augmentation tasks rather than data imputation, by generating a complete synthetic row that has been conditioned on the observed variables, the synthetic row can be utilized to impute the missing values. Since CTGAN is robust and capable of learning complex distributions, it has been utilized in the novel framework proposed in this thesis for data imputation tasks.

## 2.3 Earlier developed GAN frameworks for missing data imputation

Numerous frameworks for missing data imputation have been introduced that utilize the GAN architecture. In this subsection, well-known frameworks in this field shall be discussed. First, there is the GAIN framework[17], which the authors have shown to significantly outperform other modern data imputation methods under the missing completely at random (MCAR) assumption. Similar to the original GAN framework, the GAIN architecture consists of only one generator and one discriminator. A noteworthy difference is that the discriminator in the GAIN framework also takes a hint vector as input, which provides additional information on the likelihood that a data point is synthetically generated. Thus, the discriminator's ability to distinguish real data from imputed data increases due to this hint mechanism. As a result, the generator is forced to generate synthetic data which is even more accurate to the underlying data distribution. Based on experiments on various data sets, utilizing the hint vector boosted performance on average by 10%. Furthermore, the authors illustrated that the performance of a logistic regression classifier increased up to 79.2% after imputation with their framework instead of other imputation methods. A clinical study[5] also verified this performance boost, which showed that the GAIN framework significantly outperforms an ensemble machine learning algorithm at higher missing rates. In addition, the authors of the clinical study show that the imputation time is remarkably shorter than other imputation methods for large sample sizes and that this framework was also usable under the data missing at random (MAR) assumption. However, one of the disadvantages of the GAIN model is that training can be computationally intensive. Hence, another variation named slim GAIN (SGAIN)[9] has been proposed to create a more efficient framework. Some straightforward alterations in SGAIN are reducing the number of layers in the generator and discriminator, utilizing a different activation function, and discarding the hint vector as proposed in GAIN. A noteworthy difference is that the discriminator outputs two matrices since it is invoked twice, once for the real data (including missing values) and once for the imputed data matrix. As a result of these changes, SGAIN reduces computation time up to 30% compared to the standard GAIN framework while providing comparable model performance post-imputation. Another potential problem is that the GAIN framework does not account for individual class distributions but instead models one distribution for the whole data. This neglection for individual class distributions may negatively impact the performance when a class imbalance is present. For this reason, the CGAIN[2] was proposed. By providing the generator a label

encoding as additional input, class characteristics can be learned and used for imputations. The authors show that as the data set becomes more imbalanced, CGAIN tends to outperform the GAIN framework more stably. However, the CGAIN model increases the computational load compared to the original GAIN model.

A more sophisticated framework is the MisGAN imputer from S. Cheng-Xian Li *et al.*[4], which consists of up to three generators and three discriminators, with each having a specialized task. Firstly, a mask-generator and mask-discriminator models the missing data process. Secondly, there is the data-generator, its output is masked similarly to the actual data (with missing values), and both are used to train the data-discriminator. Lastly, there is the imputer-generator which takes as input the incomplete data cases, the mask vectors to indicate what data is missing, and a random noise vector. It then outputs completed samples, conditional on the actual observed data. These completed samples are used together with the output of the data-generator to train the imputer-discriminator. When testing the MisGAN imputer on various data sets for different missing rates, the MisGAN imputer outperforms state-of-the-art imputation methods and significantly outperforms the GAIN framework. Furthermore, according to their quantitative evaluation, the MisGAN imputer is more stable during the training phase compared to the GAIN framework. Thus, the MisGAN framework shows promising results when utilizing multiple generators and discriminators for imputation tasks. Next is the GAMIN framework[18], which has outperformed both the GAIN and MisGAN imputer at high missing rates (80% through 95%). The most notable difference compared to earlier works is that this architecture first imputes missing data with a candidate imputation, using an unconditional generator. Then, this candidate imputation is transformed by the next conditional generator into a more fitting imputation. The authors designed a confidence prediction based on how significant the difference is between the imputation proposed by the conditional generator and the proposed candidate imputation. If this difference is small, the confidence prediction shall be high. On the other hand, if the conditional generator has changed the imputation to a great extent, the confidence prediction shall be low. The authors use this confidence prediction to create a novel loss function that incorporates this confidence prediction in order to optimize the networks during training. However, the GAMIN framework underperforms compared to the previous frameworks at missing rates of 50% and below. Thus, this framework is only truly beneficial in cases of highly missing data rates.

# Chapter 3

# Theoretical Analysis

In the following subsections, the theory behind the constructed framework and its goals is elaborated to a greater extent. In addition, limitations of imputing missing values using GANs are discussed, and brief explanations are provided on how these limitations can be mitigated.

## 3.1 Problem Statement

The imputation framework aims to solve the missing data problem for a given incomplete table $T_{raw}$, with Variables V $\{V_1, ..., V_n\}$ and rows R $\{R_1, ..., R_2\}$. The set of variables V can contain a mixture of categorical- and numerical variables, making imputation more sophisticated since these different variable types need to be modeled concurrently. Firstly, the assumption is made that for any two given variables $V_a, V_b \subseteq V$, $V_a \perp\!\!\!\perp V_b$. In the case where variables do depend on each other, this can lead to insensible imputations. For instance, if $V_a$ is a start date column and $V_b$ is an end date column, and for a given row $R_i \subset R$ where $V_a$ is observed but $V_b$ is missing, the situation can occur that the $V_b$ is imputed with a value that lies before start date $V_a$. The same principle applies for the rows R in table $T_{raw}$, where rows are expected to be independent of each other. Hence, if the position of a row in table T holds predictive power, such as in a time series structure, this framework will not provide high-quality imputations as it does not take past- and future values into account.

The end goal is to create a complete table $T_{imputed}$ from a given incomplete table $T_{raw}$, where the imputed values are of high quality such that imputed rows are as indistinguishable as possible from fully observed complete rows. A prerequisite to achieving these goals is that there must be enough complete rows without missing values in table $T_{raw}$ to train multiple GAN models. The underlying CTGAN framework[16] was initially implemented for data augmentation rather than imputation. Nevertheless, the same goal can still be achieved by viewing the missing data problem as a data-generating problem. For a given row $R_i \subset R$ with values $\{x_1, ..., x_n\}$ where $x_i$ can be either observed or missing, the goal is to generate a novel complete synthetic row $R_{synthetic}$ that has been conditioned on the observed variables of $R_i$. Then, each missing value in $R_i$ is substituted with the corresponding non-missing value in $R_{synthetic}$. This process is repeated for each row in R that contains at least one missing value and ultimately results in the table $T_{imputed}$ that contains no missing values and is ready to be used for further analysis.

In order to create high-quality imputations, a generator G and discriminator D are trained by competing against one another in a mini-max competition. G aims to minimize the function below while D, on the other hand, tries to maximize it.

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$

Where D(x) stands for the probability that x came from the training data instead of the output of G. Optimally, a global optimum is reached after enough training iterations where the Generator is able to mimic the distribution of the real data as closely as possible. Thus, samples from the Generator's distribution can then be drawn, which are indistinguishable from samples drawn from the actual trained data.

## 3.2 Foreseeable Limitations

Although the idea is to generate a complete row $R_{synthetic}$ that has been conditioned on all the observed variables in $R_i$, this goal is often not achievable in practice. The framework aims to generate synthetic rows by sampling from a joint conditional distribution, where sampling has been successful if the synthetic row has the same values for all the observed variables in $R_i$. In the case that a row has been sampled that does not meet all the conditions of the observed variables in $R_i$, the generated row is rejected. Thus, generating synthetic rows works through an elimination-based sampling process. If there are many observed variables to condition on, the model can end up rejecting all synthetic rows because it cannot meet all the specific conditions. Hence, a solution has been implemented that deals with the situation where no synthetic row could be generated. The solution works by first sampling multiple synthetic rows under fewer conditions and then finding out which of these synthetic rows is most comparable to the observed variables in $R_i$. A more detailed elaboration of this solution is given in the next chapter.

The following limitation is that GAN models can have convergence difficulties during the training phase. If one of the neural networks learns too quickly, the learning capabilities of the adversarial neural network will suffer. However, the training stability can be improved by considering the two networks' learning rates. Furthermore, an often occurring problem for GANs is mode collapse. Hence, after the training has finished and the missing values have been imputed, the generated values are analyzed to detect whether mode collapse has occurred through visual inspection. This framework's last challenge is the computational time complexity of fitting multiple GAN models and creating multiple synthetic rows for one missing value. Nonetheless, The severity of the time-complexity problem depends on the number of rows that require imputation and the size of table $T_{raw}$. In addition, the performance of having multiple GAN models instead of one is analyzed to determine whether the possible performance gain justifies the additional computational load. Nevertheless, multiple interventions have been made to increase the time efficiency of the proposed framework.

# Chapter 4

# The Proposed Framework

In this chapter, the proposed framework will be explained in greater detail alongside its specific components. A high-level overview of the framework has been shown in figure 4.1. Firstly, the framework takes as input a raw data set with missing values (indicated by 'NA'), where the variables V can be either categorical or continuous. Note that in the example matrix shown in the high-level overview, there are three fully observed rows since training is not possible if no complete rows exist in the original input matrix. As long as the input matrix is not empty, contains complete rows, and also contains rows with missing values, it is eligible to continue to the following steps. Once these requirements have been checked, the data types of each of these variables are determined automatically. Afterward, the input matrix is normalized and remapped such that all continuous variables (including date times) fall in the same range of [0,1]. For categorical variables, each distinct value is given a different integer number. Therefore the range will be between 0 and the number of distinct variables. After the pre-processing steps have finished, the data format is now suitable for the training phase since it only contains numerical values at that point. The data set is first divided into two matrices, one which contains no missing values while the other contains at least one missing value for each row. The fully observed matrix is used to train each of the three different CTGAN models. Each trained model is then used to impute the missing values in the incomplete matrix, and the final imputations are then chosen based on a majority vote. Lastly, the matrix is denormalized, and categorical variables are remapped again so that the data format is identical to the input matrix. As an end product, the researcher obtains a complete matrix where each missing value has been imputed, and all the observed variables remain exactly the same. In the following subsections, each component is elaborated on in more detail.

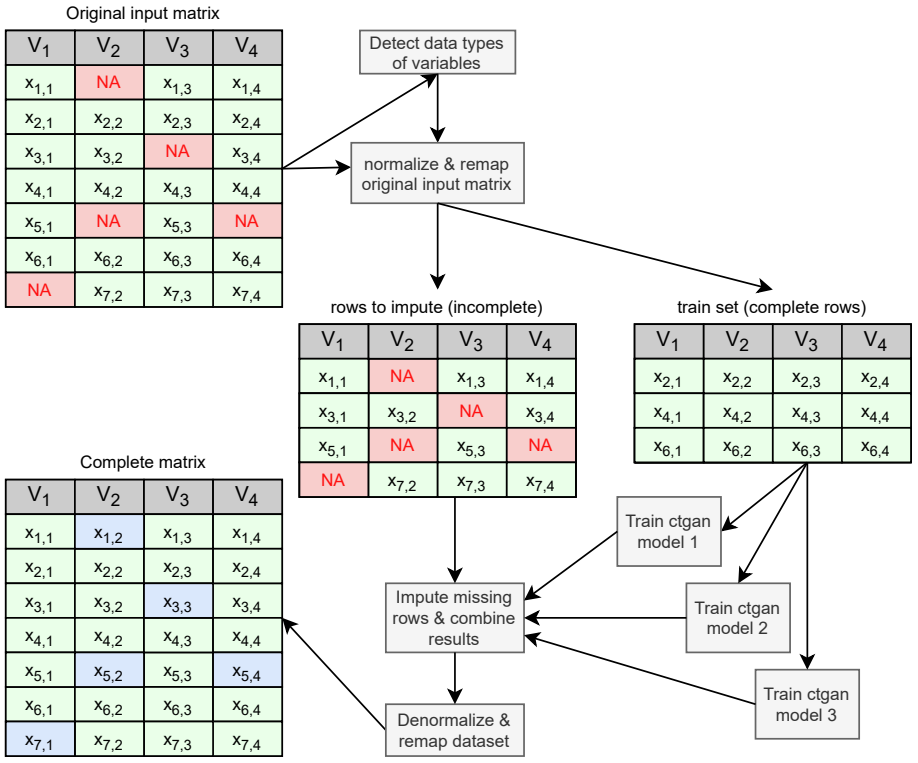


Figure 4.1: High level workflow of the steps taken in the proposed framework

## 4.1 Data pre-processing

Firstly, the data types of the variables in the input matrix need to be determined so that they can appropriately be transformed to suit the required data format when training the CTGAN models. In total, there are three different data types that the framework differentiates from each other: Continuous variables, categorical variables, and date-time variables. Notable is that date-time variables require an additional pre-processing transformation, hence are not given the same data type as other continuous variables. In the first step, the value formats for all variables are checked to see whether they fit a date-time format such as 'year-month-day.' If all the values in the variable have this same date-time format, then the date-time data type is chosen for this variable. Another simple situation is when the values of a variable contain alphabetical characters, in which case the variable is categorical. More problematic is that variables that contain numerical values can not always be classified as continuous variables. For instance, take a binary variable where values can either be 0 or 1; this should be classified as a categorical variable. However, if the number of distinct values is greater while the number of rows in the matrix is small, this might indicate that the variable type is continuous. While on the other hand, if the number of distinct values is larger while the number of rows in the matrix is significant as well, this could indicate that the variable type is categorical. Thus, some threshold is necessary to determine the cut-off point based on a proportional measurement that considers the variable's distinct values and the number of rows in the input matrix. This threshold can simply be calculated as:

$$distinct\_ratio\_threshold = \#distinct\_values \div \#rows \tag{4.1}$$

where:

$\#distinct\_values$ = number of unique values for a given variable
$\#rows$ = number of rows in the input matrix

Since the number of distinct values can never surpass the number of rows, the threshold shall lie between 0 and 1. To set the distinct ratio threshold, the researcher can pass a fraction in this range as a hyperparameter in the proposed framework. Take the following illustration of the effect that the distinct ratio threshold can have on the definitive chosen data type: For a given variable, the number of distinct values is 20, and the number of rows in the matrix is 1000, then the distinct ratio of that variable is 0.02. Assuming that the threshold is set to 0.01, the variable will be classified as a continuous variable. However, if the number of rows in the matrix is greater than 5.000 and the number of distinct values remains 20, then the distinct ratio is smaller than 0.01. Therefore, the variable will be classified as a categorical variable.

Once the data types have been detected for all variables in the input matrix, the next step is to transform the matrix. Each variable is transformed, so the continuous variable falls in a range of [0,1], and categorical variables are mapped to an integer number. Date-time variables first have to be recast into a long integer and afterward similarly transformed to fall in the same range of [0,1]. It is necessary that continuous variables have the same range since comparing variables to each other would otherwise yield unfair results. Take for instance the situation where variable $V_1$ has a range of [0,10] and variable $V_2$ has a range of [0,10.000]. A small prediction error in $V_2$ will likely be bigger than the whole range of variable $V_1$; therefore, $V_2$ would dominate the evaluation metrics unless they are both transformed into the same range. This transformation can simple be performed by applying the following formula to each value in the continuous variable:

$$new\_value = (current\_value - min) \div (max - min) \tag{4.2}$$

where:

$min$ = lowest value in the variable
$max$ = largest value in the variable

This assumes that the number of distinct values in the continuous variable is $\geq 2$, such that $max \neq min$. Which ultimately results in a new set of values that fall in the range of [0,1]. Next, the categorical variables need to be remapped to numerical values. Otherwise, the data format does not meet the requirements to train a neural network. One standard method to transform categorical data into numerical variables is to perform one-hot encoding. For a given categorical variable $V$ with n distinct categorical values, n new binary variables are created to indicate the categorical value. However, this can drastically increase the input matrix size with a tremendous amount of redundant information, especially if the number of distinct values is significant. This size increase will make the training phase of the CTGAN models significantly slower; hence, a simple alternative method has been chosen. Each distinct value in the categorical variable $V$ is remapped to an integer number. Thus, the range of the variable will be between 0 and the number of distinct values. These mappings are saved such that they can be reverted to the original values after imputation. This remapping strategy is a suitable solution since the model does not yield fractional imputations for these variables, which would be insensible since the categorical values have no order. After the normalization and remapping have finished, this should result in a data set with numerical- and missing

values. The complete rows are then split from rows that contain missing values such that they can be used for the training phase, which shall be discussed in the next section.

## 4.2 CTGAN training

The training aspect of the GAN model is of vital importance for successful high-quality data imputations. Figure 4.2 illustrates the training of the generator and discriminator for one training iteration.

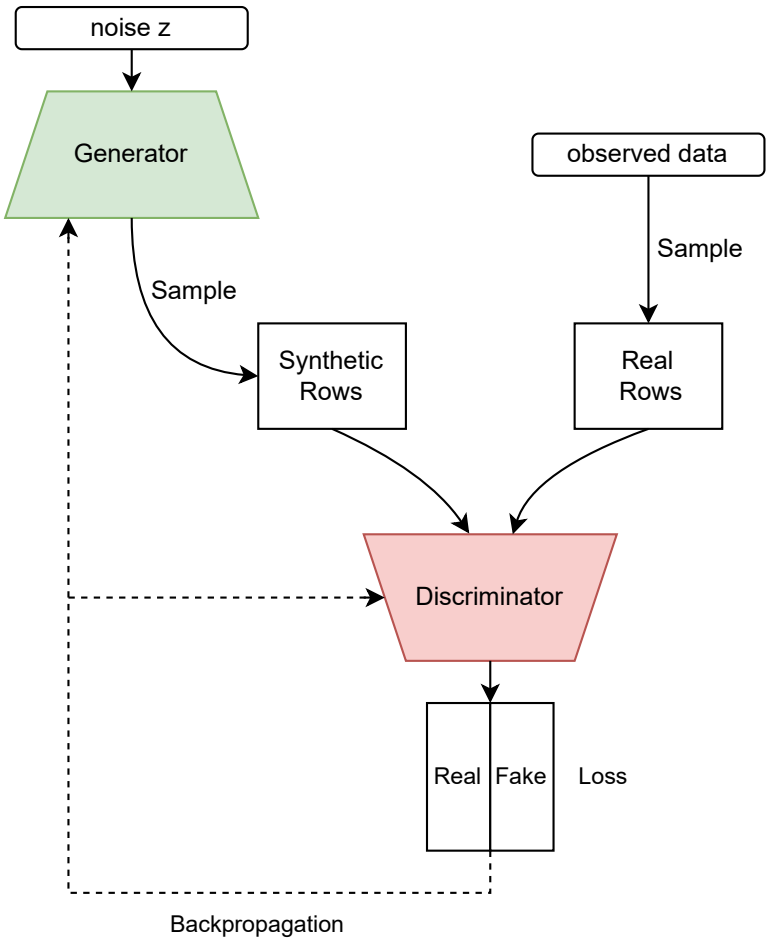


Figure 4.2: Overview of the GAN learning methodology, for one training iteration

A sample is taken from the real data and from the generator's distribution; both samples are then forwarded to the discriminator. The discriminator does not know which rows originate from the observed data or the generator. Its goal is to calculate the probability of the row originating from the real observed data for each sampled row. The discriminator uses the probability to classify rows as fake or real, from which the loss is calculated. At the beginning of training, the samples produced by the generator are likely not of high quality. Nevertheless, using backpropagation of error, the weights of the neural networks are updated after each training iteration. As a result, after several training iterations, the samples produced by the generator will more closely resemble samples from the observed data. Although, this assumes that the training phase has been successful, which often is not the case. For instance, the generator can get too strong compared to the discriminator, resulting in the generation of low-quality data. On the other hand, if the discriminator overpowers the generator, then the generator might eventually fail to learn how to produce more realistic outputs. Even though the networks compete against one another by maximizing the loss of the other, it is still essential that both networks converge to a balance.

Parameter tuning can help to achieve a more stable training phase. However, many parameters can be tweaked, such as the batch size and the number of discriminator updates in relation to the generator. Though, the number of training iterations and the learning rates of the generator and discriminator are more influential for the overall training stability of the two neural networks. In figure 4.3 a failure case of GAN training is illustrated and afterward explained.

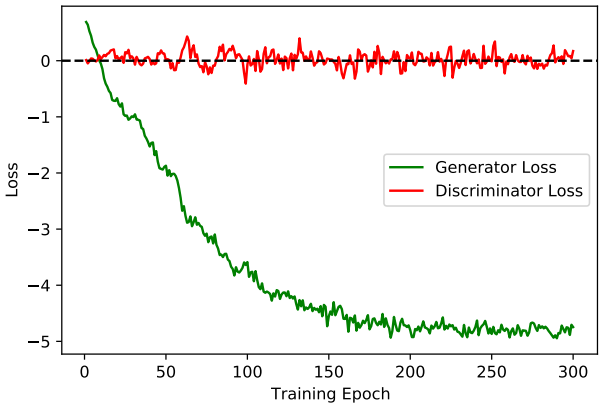


Figure 4.3: Generator and discriminator loss per epoch on COMPAS data set. The learning rate of the Generator has been set to 2e-4, and the discriminator to 2e-5.

In the training example above, the discriminator learning rate has been reduced by tenfold compared to the generator. As a result, the generator loss decreases steadily after each epoch while the discriminator loss remains relatively stable around 0. Thus, the generator is increasing in strength, which results in an overall strength imbalance between the generator and discriminator. Since the discriminator is too weak due to the lower learning rate, the data quality of the synthetic data created by the generator will likely be of poor quality. Hence, it is necessary that a balance is reached between the two networks. The goal is to prevent the two networks from diverging and instead have them converge to each other such that they are roughly equal in strength. In figure 4.4, multiple combinations of learning rates are shown and how they affect the training stability of the GAN model.

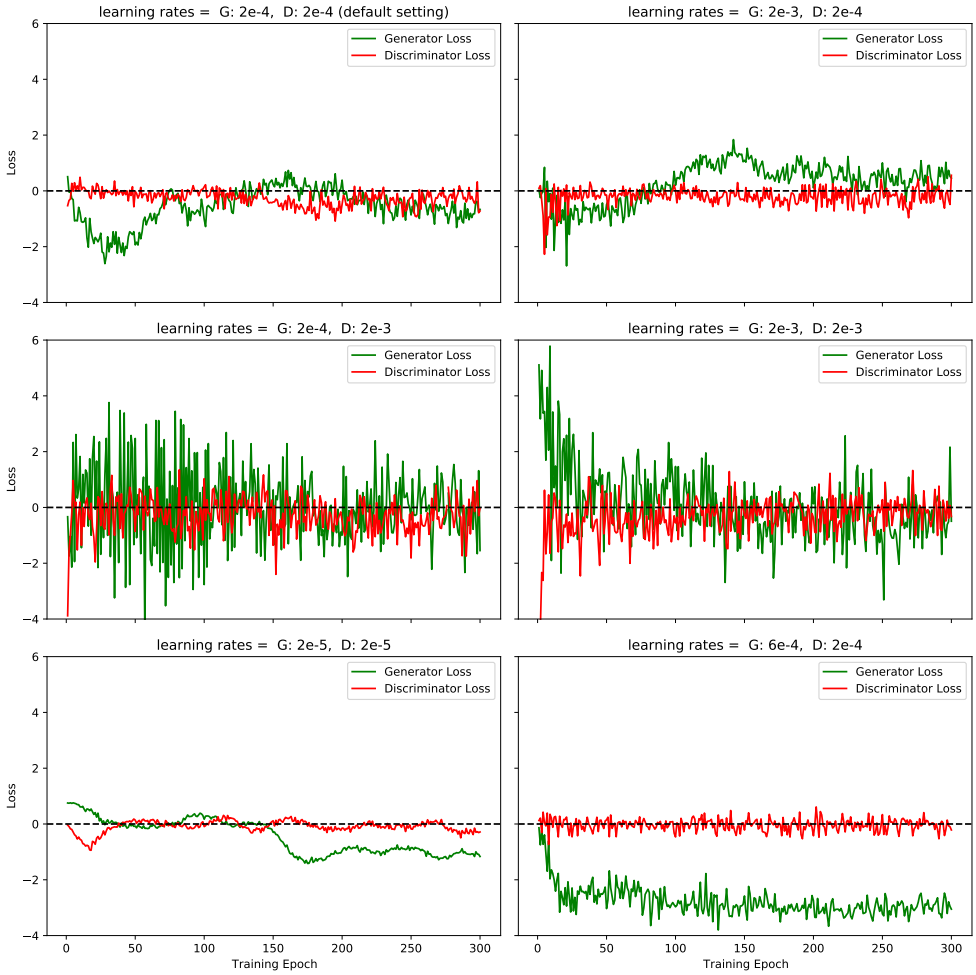


Figure 4.4: Generator and discriminator loss per training epoch for the COMPAS data set, using various combinations of learning rates.

Each model has been trained for 300 epochs in the six illustrations above. Also, note that the range loss is the same for all six plots since this makes them easier to compare. By default, the learning rates of the generator and discriminator are set to 2e-4. Interestingly, the generator diverges for the first 50 epochs before converging into a balance with the discriminator. Therefore, emphasizing the necessity that the networks are trained for a sufficient amount of training iterations. In this case, the default learning rates were sufficient since the generator and discriminator stabilized on their own in the second half of the training. Increasing the learning rate of the generator while leaving the learning rate as the default value resulted in contradicting results. In the top-right, the generator's learning rate is tenfold higher than the generator's, while in the top-bottom plot, the generator's learning rate is threefold higher. Noteworthy is that convergence was achieved with a tenfold increase but not a threefold increase. In the middle two plots, the learning rate of the discriminator has been increased tenfold. However, increasing the discriminator's learning rate does not seem to benefit the overall learning stability. As the plots show, the loss of both networks becomes too volatile

during training. Lastly, both learning rates are drastically reduced on the bottom left plot. Nevertheless, this resulted in failure since the generator overpowered the discriminator in the second half of training. Thus, the learning rates can drastically affect the training stability and might need to be slightly altered based on the training data. Hence researchers should pay close attention to the loss during training and might need to experiment with various settings to achieve a balance between the networks.

## 4.3 Imputation method

Once all three models have been correctly trained, it is time to perform the last step of imputing the missing values. Optimally, the goal would be to condition on all observed variables for an incomplete row when generating a synthetic row. This optimal scenario has also been illustrated in figure 4.5, where the variable $V_2$ is missing and synthetic rows are generated conditional on $\{V_1, V_3, V_4\}$. Unfortunately, this might not always be achievable if there are many conditions or when the conditions contain precise numerical values. The reason for this is that conditional sampling works through a rejection-based process. The trained generator is asked to create a synthetic row, and only if the synthetic row has the same values for all specified conditions then this synthetic row shall be accepted. However, if the sampled synthetic row does not meet all specific conditions, it will be rejected. The process is repeated multiple times until a row has been sampled that meets all the requirements or a maximum number of tries has been reached. This may be problematic because if a sample could not be generated that meets all the conditions before the maximum number of tries has been reached, the missing values in that row can not be imputed. Therefore, defeating the proposed model's purpose, which would take an incomplete data set as input and needs to return a completed data set. A solution has been implemented in the proposed framework to ensure that a complete data set is always returned. This solution is elaborated on in detail later in this section.

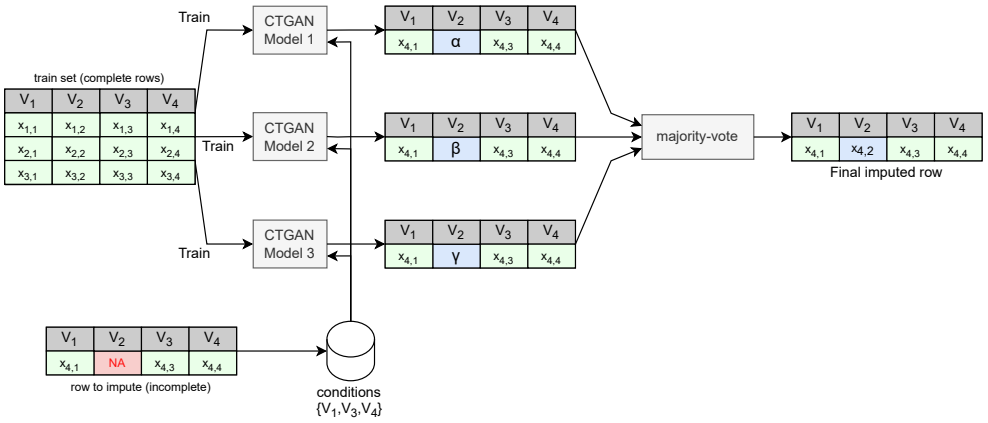


Figure 4.5: high level imputation overview for imputing one missing value

When diving back into figure 4.5, we can see that the proposed framework creates three candidate imputations for each missing value. These candidate imputations are indicated by the Greek letters $\alpha$, $\beta$, and $\gamma$. The framework determines a final imputation based on a majority vote. In the case that all three candidate imputations have the exact same value, any of the three can simply be imputed. If on the other hand, $\alpha = \beta \wedge \alpha \neq \gamma$ or $\alpha = \gamma \wedge \alpha \neq \beta$, then $\alpha$ can be chosen as the final imputation. If $\beta = \gamma \wedge \alpha \neq \beta$, then $\beta$ is chosen as

the final imputation. More difficult is when $\alpha \neq \beta \neq \gamma$, because in this case the best candidate imputation needs to be determined. One possible way is to evaluate how closely the imputed row resembles the complete rows used to train the GAN models. The evaluation function yields a number between 0 and 1, based on how closely the synthetic row resembles the statistical properties of the original data. The candidate imputation that achieves the highest score can then be used as a final imputation. However, this can be a computationally expensive task, while the proposed framework already has a high computational load. Hence, it is more practical to randomly choose any of the three candidate imputations as the final imputation.

Next, the implemented solution shall be discussed, which deals with situations where a synthetic row could not be generated when conditioned on the observed variables. See Algorithm 1 for the pseudo-code of the implementation in the proposed framework for one incomplete row. As stated earlier, optimally, we would generate a synthetic row conditional on all observed variables. However, this can often not be achieved when there are many variables to condition on or when the variable is continuous and thus has many distinct values. The proposed solution is to generate multiple synthetic rows that have only been conditioned on the observed categorical variables. Then, the synthetic rows that most closely resemble the observed continuous variables in the original row are established. This resemblance is determined using the euclidean distance metric, where a shorter distance implies a closer resemblance to the continuous variables in the original row. For two continuous column vectors $\alpha, \beta \in \mathbb{R}^n$, the euclidean distance can simply be calculated as:

$$Euclidean\_distance = \sqrt{\sum_{i=1}^{n}(\alpha_i - \beta_i)^2} \tag{4.3}$$

As an illustration, take a set of continuous column vectors $[V_1, V_2, V_3]$. $\alpha$ = [1,5,3] and $\beta$ = [0,4,3], then the Euclidean distance between these vectors is $\sqrt{(1-0)^2 + (5-4)^2 + (3-3)^2}$ which equals 1.414. Since up to 50 synthetic rows are sampled conditional on the categorical variables, this Euclidean distance is likewise calculated up to 50 times between the original row's numerical variables and each synthetic row's numerical variables. Then, the top three synthetic rows with the shortest distance are used for a majority vote to pick the final imputation from this one model. If all three synthetic rows have different candidate imputations, the row with the shortest distance is chosen as the final imputation. In the other cases, the majority vote applies similarly to figure 4.5. Nevertheless, conditioning on all categorical variables might still fail when no synthetic row was sampled with the same values. Hence, the following process is applied to ensure that a complete data set can still be provided. Firstly, we try to condition on all categorical variables. However, if sampling was unsuccessful, then one of the conditions is removed randomly, and sampling is retried. This process is repeated until sampling is successful and the missing values have been imputed, or until no conditions are left, after which no samples are rejected.

**Algorithm 1** Missing row imputation algorithm

**Input:**
    row *row with missing values to impute*
observed_variables: *Save all non-missing variables of input row in a dict*
cont_vars: *Save all observed continuous variables of the input row*
conditions: *From observed_variables save categorical variables*
**while** *#conditions* $\geq 0$ **do** *'#' implies 'number of'*
    **try:**
        new_data = *model.sample(conditions, nrows=50)* *Sample up to 50 rows*
        distances = [ ] *List to save distances from real row to synthetic rows*
        **for** fake_row **in** new_data **do**
            dist = euclidean(fake_row[cont_vars], real_row[cont_vars]))
            distances.append(dist)
            *save euclidean distance between the rows*
        **end**
        sort(distances) *sort distances list in ascending order*
        first_place_fake_row = distances[0]
        second_place_fake_row = distances[1]
        third_place_fake_row = distances[2]
        **for** variable **in** row **do**
            **if** row[variable] = NaN **do** *'NaN' implies a missing value*
                *Only the missing values are imputed in the original row*
                $\alpha$ = first_place_fake_row[variable]
                $\beta$ = second_place_fake_row[variable]
                $\gamma$ = third_place_fake_row[variable]
                **if** $\alpha \neq \beta$ and $\beta = \gamma$ **then**
                    row[variable] = $\beta$ *impute missing value with $\beta$*
                **else** *Impute based on majority vote.*
                    row[variable] = $\alpha$ *impute missing value with $\alpha$*
        **end** *Imputations done, thus while breaks in line 36*
    **except:** ValueError *Thrown if in line 8 no samples are generated*
        **if** *#conditions* $\geq 1$ **do**
            conditions.pop(random)) *remove one random condition & retry*
            **continue** *We keep trying with one less condition until successful*
        **else**
            **break**
    **break** *Break from while loop & redo process for every incomplete row*
**end**

The imputation process can be computationally costly, depending on the number of missing values and conditions. Additionally, there is a higher chance of rejection for a sampled row if the condition is based on a categorical variable with many distinct values. On the other hand, if the categorical variable only contains two groups and is not highly imbalanced in the data set, then the generator easily meets this condition. Figure 4.6 illustrated the computation time in seconds for 100 imputations with a growing amount of conditions.

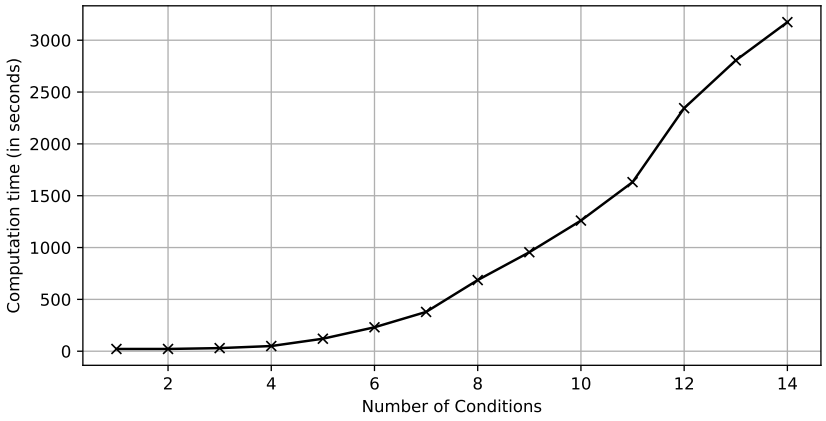


Figure 4.6: Computation time of imputing 100 missing values for one underlying GAN model on the Adult Income data set.

As can be deduced from the figure, imputation was finished within one minute when less than five conditions were used. However, the imputation time increased to almost an hour when the number of utilized conditions was equal to fourteen. Note that this does not mean that synthetic rows always managed to condition on these fourteen variables. Instead, sampling often had to be re-tried with fewer and fewer conditions until successful. The researcher can indirectly affect the number of conditions by fine-tuning the distinct ratio threshold during training. Unfortunately, there is not a one-size-fits-all for the best number of conditions to use since this heavily relies on the actual input data. Hence, researchers should consider the number of missing values to impute and the difficulty of their conditions before imputation. Nevertheless, if the time constraint is not an issue, a high number of conditions could still be used without having to take into account the above-mentioned considerations.

# Chapter 5

# Experimental Evaluation

In this section, the proposed framework will be evaluated in two different ways for three different non-simulated data sets. Firstly, a novel evaluation metric is constructed that yields an overall prediction error while considering mixed data type variables for fair comparisons. Secondly, the performance of a machine learning classifier is evaluated post-imputation and compared to other imputation frameworks. However, a brief overview of the utilized data sets is first given, and their missing value patterns are analyzed. Afterward, the two evaluation metrics shall be discussed in greater detail and then applied to the proposed framework and various other widely-used imputation frameworks.

## 5.1 Utilized data sets

The performance of various imputation frameworks has been evaluated on three different real-world data sets. In this section, the data sets are described, and the missing values patterns are analyzed to check whether they are suitable for data imputation or instead need to be slightly altered pre-imputation.

| raw data set summary | | | | |
|---|---|---|---|---|
| **Dataset** | **#rows** | **#Attributes (Contin., Categ.)** | **Target-Class Proportions** | **#Incomplete-rows** |
| German | 999 | 21(3,18) | Good credit(70%), Bad credit(30%) | 567 |
| Compas | 7214 | 52(22,30) | Recidivist(45%), non-recidivist(55%) | 7214 |
| Adult | 48842 | 15(5,10) | high-income(24%), low-income(76%) | 3620 |

Table 5.1: Brief overview of the characteristics of each utilized data set

As can be deduced from table 5.1, the three data sets have a significant size difference compared to each other. This difference can be beneficial, as this allows for better insights into the effect of the training sizes on the data imputation quality from the proposed framework. All three data sets contain both continuous- and categorical attributes. However, the Compas

data set stands out because of its many features. Furthermore, large class imbalances can be identified from the German credit- and Adult income data sets. This imbalance can be problematic when evaluating the performance gain of a classification model post-imputation compared to other frameworks, which is what the second evaluation metric is based upon. Hence, the class imbalance needs to be mitigated to ensure reliable results. Lastly, a noteworthy takeaway from the data set characteristics overview is that the number of rows for the Compas data set equals its number of incomplete rows. Thus, this data set contains not one complete row while this is a prerequisite, as stated in the theoretical analysis chapter. In the following subsections, the missingness patterns shall be further analyzed for each data set, and a solution to the aforementioned issue shall be elaborated.

### 5.1.1 German Credit Data set

In this data set, the rows represent people that applied for a credit loan and, based on multiple features, are classified as good (low-risk) or bad (high-risk). As mentioned earlier in table 5.1, there is a class balance of 70% for the low-risk class and 30% for the high-risk class.

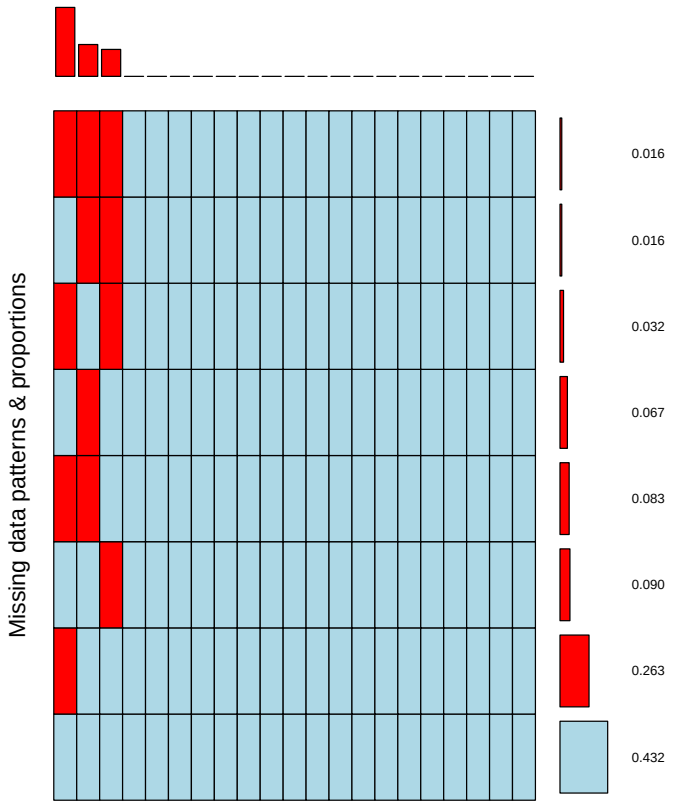


Figure 5.1: Missing data patterns of German Credit data set, where the light-blue cell indicates observed data, and red cell indicates a missing value.

In figure 5.7 an overview is provided of all the unique missing patterns in the data set. On the right, the proportions indicate the commonness of the corresponding missing pattern. Each column indicates a variable in the data set; a cell is colored red if the value is missing for that variable or blue if observed. On the top, a histogram of the proportion of missing values is provided per feature. Although, without a y-axis from which the specific proportions can be deduced. The most important takeaway from the figure is that there are 432 fully-observed rows. A fully-observed data set could also be obtained by removing the three columns containing missing values. Nevertheless, for the first evaluation metric, 432 rows are used, of which 50 are reserved for the test set. The small training set of this data set allows for possible insights into the performance of the proposed framework when the generator and discriminator are trained using a variety of training set sizes.

### 5.1.2 COMPAS Recidivism Data set

The compas data set originates from the popular compas algorithm, which has been used in the American judicial system to determine the risk of a defendant re-offending a crime within two years. The data set contains information such as criminal history, jail time, ethnicity, and other sensitive information. Based on these features, a risk score is calculated by this algorithm, with the end goal of mitigating human bias in the judicial system. However, the algorithm had the opposite effect since it instead increased bias by assigning higher risk scores based on specific ethnic groups. In figure 5.2 the unique missing patterns are illustrated alongside the proportion of their presence in the data set.

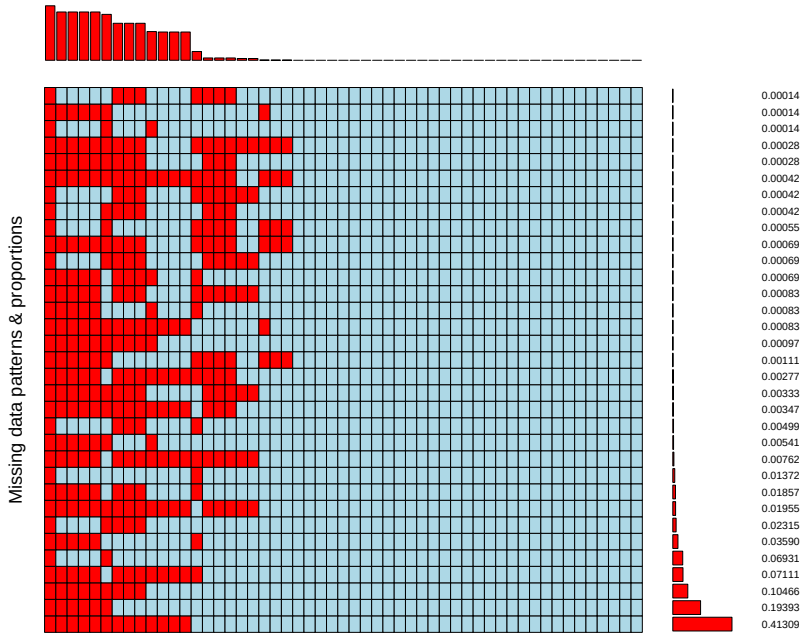


Figure 5.2: Missing data patterns of Compas Recidivism data set, where the light-blue cell indicates observed data, and red cell indicates a missing value.

Although the data set contains 7214 rows, there are no complete rows when examining the missing patterns. This is the consequence of a subset of variables with a high proportion of missing values. There is even one column that has no observations at all. Hence, these columns are removed so there are enough complete rows to perform the analysis. After removing these variables, there are 6951 complete rows which are then divided into a training set and a test set.

### 5.1.3 Adult Income Data set

Lastly, the adult income data set contains 48842 rows, of which 45222 are completely observed. Each row represents a person and contains sensitive information such as their ethnic background, gender, and education level. The target outcome is a binary variable that describes the person's income, specifically whether the income is greater than or equal to 50.000 or lower than 50.000. In Figure 5.9, it is clear that a large proportion of the rows in this data set is completely observed (92.58%), which is more than sufficient to split into a train- and test set for both evaluation metrics. However, as becomes clear from table 5.1, the target variable has a very large class imbalance that must be dealt with for the second evaluation metric. Nevertheless, the large amount of fully-observed rows should not pose a problem when under-sampling from the majority class.

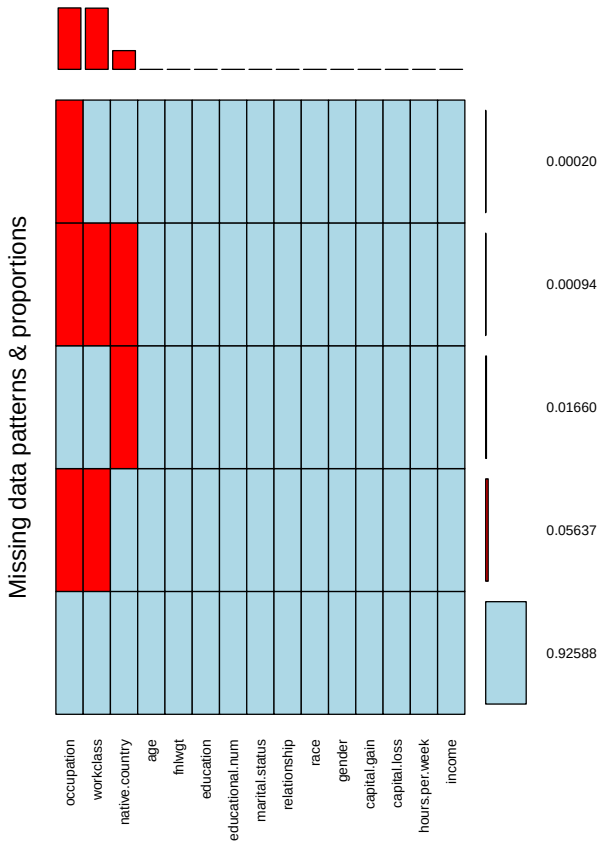


Figure 5.3: Missing data patterns of Adult Income data set, where the light-blue cell indicates observed data, and red cell indicates a missing value.

## 5.2 Evaluation metrics description

### 5.2.1 Custom prediction error formula

The first evaluation metric has been designed to reflect the imputed values' overall prediction error. Traditional evaluation metrics such as (root) mean squared error or mean absolute error are, in most cases, inapplicable since the provided table $T_{raw}$ can contain a mixture of categorical- and numerical variables. If numerical variables in $T_{raw}$ have large range differences, this could result in unfair comparisons between these variables. To clarify, a small prediction error on a variable with a large range will likely dominate over a big prediction error on a variable with a small range. It is for this reason that all numerical variables are normalized to fall in the range of [0,1]. The prediction error of an imputed value for these variables can then simply be calculated as $\mid x_{predicted} - x_{observed} \mid$.

For categorical variables, the prediction error of a missing categorical value $x_i$ is either 0 if $x_{i,predicted} = x_{i,observed}$, or 1 if $x_{i,predicted} \neq x_{i,observed}$. Thus, the assumption is made that for any categorical variable that differs from the observed variable, the imputed variable is not worse than any other imputed categorical variable. This assumption might not always be true, but implementing a different system to determine different weights if $x_{i,predicted} \neq x_{i,observed}$ depending on the imputed value might not be feasible without additional information about the variable from the user. Hence, the trivial solution has been chosen to regard all imputed categorical variables as equally wrong if not the same as the observed value. Furthermore, combining the categorical- and numerical prediction errors into one score can be problematic, even though both fall in the same range of [0,1]. Namely, the imputations of the categorical variables are seen as either right or wrong. In contrast, the imputations of the numerical variables are evaluated based on the distance from the observed variable. As a result, the average prediction error of the categorical variables will likely be higher than that of the numerical variables. In order to prevent the average categorical prediction error from dominating the overall prediction error score, an $\alpha$ constant is calculated to scale the average prediction error such that both categorical- and continuous variables equally contribute to the overall prediction error. The $\alpha$ constant is determined by first calculating the average of average-pairwise distances for all numerical variables, dividing that by the average of average-pairwise distances for all categorical variables. Thus, if these distances for all numerical variables are smaller than the distance for the categorical variables, $\alpha$ shall be less than one such that both equally contribute to the prediction error. Formally, the overall prediction error can be described as:

$$
\begin{aligned}
PredictionError = \frac{\#numeric}{\#total} \times \sum_{i=1}^{n_{num}} (|x_{i,observed} - x_{i,predicted}|) \\
+ \frac{\#categoric}{\#total} \times \alpha \times \sum_{i=1}^{n_{cat}} \left( \begin{cases} 0, & \text{if } x_{i,observed} = x_{i,predicted} \\ 1, & \text{if } x_{i,observed} \neq x_{i,predicted} \end{cases} \right)
\end{aligned}
$$

where:

$\#numeric$ = number of numerical variables in $T_{raw}$
$\#total$ = total number of variables $T_{raw}$
$\#categoric$ = number of categorical variables in $T_{raw}$
$n_{num}$ = number of missing numerical values in $T_{raw}$
$n_{cat}$ = number of missing categorical values in $T_{raw}$

This prediction error is calculated under an increasing missing rate for different imputation frameworks. A lower prediction error would imply that, on average, the imputations are more accurate compared to the observed variables. The results are specified and further elaborated in subsection 5.3.

### 5.2.2 Classifier performance post-imputation

The second methodology to evaluate the imputation quality of the proposed framework is to determine the performance difference of a classification model post-imputation when imputed with different frameworks. In this case, a random forest classification model will be used, and the performance is based on the accuracy- and ROC-AUC scores. For a given data set, 80% of the data will be used for training data and the remainder 20% for the test set. No validation set is reserved since hyperparameter tuning is not essential for a random forest model, unlike other classification models, such as k-nearest neighbors. Firstly, imputation models will be trained on the complete rows of the 80% training set and afterward used to impute all the missing values such that the training set contains no missing values. Then, the random forest model shall be trained on this complete training set and used to classify the binary target variable for all the rows in the test set. After the classification has been performed, the accuracy- and ROC-AUC score can be calculated for the random forest model and compared to the other frameworks. In order to handle class imbalances in the utilized data sets, rows from the majority class shall be undersampled to create a balanced training set. The accuracy score of a classification model can formally be described as:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.1}$$

The ROC-AUC score indicates the class separability capacity of the binary classification model. When the AUC is 1, this will imply that the model perfectly distinguishes between both classes. On the other hand, if the AUC score equals 0.5, the model has comparable performance as a random classifier. The proposed framework will be compared to the same imputation frameworks as in the first evaluation metric. After imputation of the missing data in the training set, for each imputation framework, the random forest model accuracy- and ROC-AUC scores will be calculated. The framework with the highest impact on the random forest performance is chosen as the better imputation framework in this setting.

## 5.3 Evaluation results

### 5.3.1 Results of custom prediction error

| | Missing rates in German Credit Data | | | | |
|---|---|---|---|---|---|
| **Framework** | 10% | 30% | 50% | 70% | 90% |
| TGAIN (1 model) | 0.2487 | **0.2543** | **0.2557** | 0.3176 | 0.3197 |
| TGAIN (3 models) | **0.2416** | 0.2605 | 0.2685 | **0.2941** | **0.2948** |
| MissForest | 0.3250 | 0.3187 | 0.3140 | 0.3527 | 0.3980 |
| MICE | 0.2840 | 0.3140 | 0.3054 | 0.3084 | 0.3002 |
| KNNImputer | 0.2525 | 0.2941 | 0.2941 | 0.3290 | 0.3155 |
| IterativeImputer | 0.3335 | 0.3368 | 0.3581 | 0.3572 | 0.3412 |

Table 5.2: Results are based on 382 training rows and 50 test rows, a lower prediction error implies better overall imputations

| | Missing rates in Compas Recidivsm Data | | | | |
|---|---|---|---|---|---|
| **Framework** | 10% | 30% | 50% | 70% | 90% |
| TGAIN (1 model) | 0.1521 | **0.1532** | 0.1537 | 0.1558 | 0.1690 |
| TGAIN (3 models) | 0.1492 | 0.1582 | **0.1403** | **0.1446** | **0.1604** |
| MissForest | **0.1455** | 0.1569 | 0.1583 | 0.1662 | 0.1981 |
| MICE | 0.1618 | 0.1672 | 0.1600 | 0.1881 | 0.2007 |
| KNNImputer | 0.1535 | 0.1536 | 0.1821 | 0.1665 | 0.1821 |
| IterativeImputer | 0.1692 | 0.2148 | 0.1841 | 0.1841 | 0.3197 |

Table 5.3: Results are based on 6801 training rows and 150 test rows, a lower prediction error implies better overall imputations

| | Missing rates in Adult Income Data | | | | |
|---|---|---|---|---|---|
| **Framework** | 10% | 30% | 50% | 70% | 90% |
| TGAIN (1 model) | 0.1279 | 0.1236 | **0.1197** | 0.1263 | 0.1585 |
| TGAIN (3 models) | 0.1208 | 0.1225 | 0.1215 | **0.1247** | **0.1557** |
| MissForest | 0.1297 | **0.1191** | 0.1289 | 0.1328 | 0.1610 |
| MICE | **0.1148** | 0.1276 | 0.1315 | 0.1303 | 0.1591 |
| KNNImputer | 0.1310 | 0.1313 | 0.1375 | 0.1536 | 0.1639 |
| IterativeImputer | 0.1317 | 0.1437 | 0.1436 | 0.1613 | 0.1620 |

Table 5.4: Results are based on 45722 training rows and 500 test rows, a lower prediction error implies better overall imputations

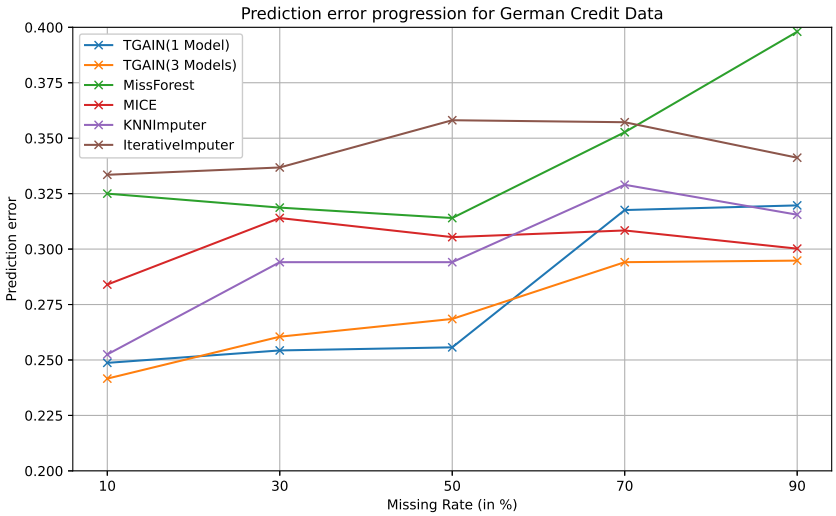


Figure 5.4: Prediction error progression of the different imputation frameworks for the German Credit data set, a lower score implies better overall imputations.

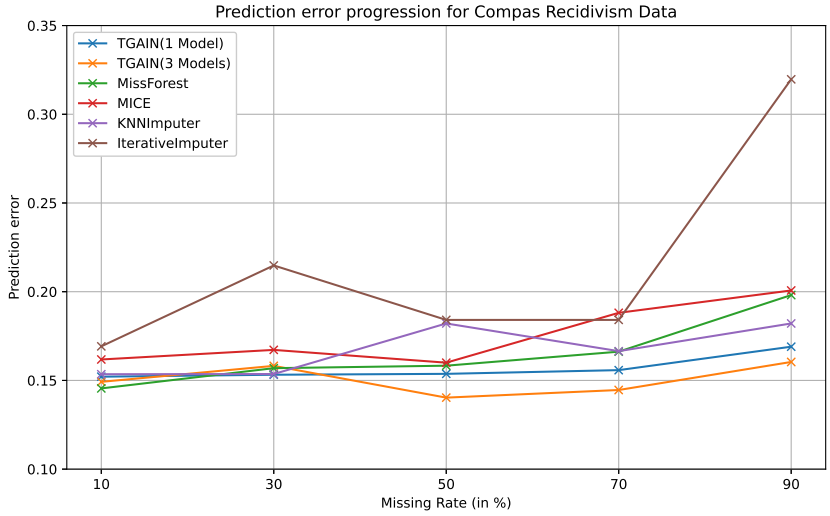


Figure 5.5: Prediction error progression of the different imputation frameworks for the Compas Recidivism data set, a lower score implies better overall imputations.

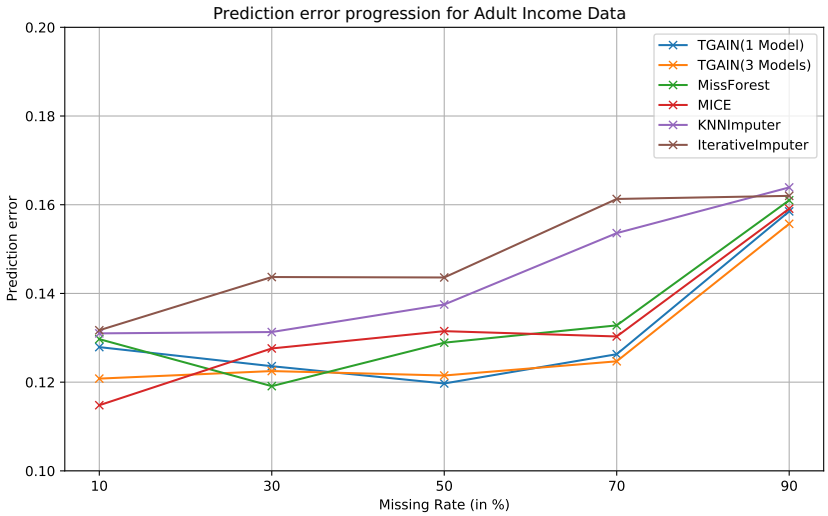


Figure 5.6: Prediction error progression of the different imputation frameworks for the Adult Income data set, a lower score implies better overall imputations.

The proposed framework (TGAIN) has been compared to four other state-of-the-art imputation frameworks and also a less computational expensive variation that uses one GAN model instead of three. In addition, the evaluation has been performed on three data sets ranging from a low amount of training data (382 rows) to a high amount of training data (45722 rows). When examining the results of the German Credit data set, the proposed framework achieved the lowest prediction error scores across the whole missing rate spectrum from 10% up to 90%. Also, when utilizing three GAN models, the prediction error increased stably as the missing rate increased. This stable increase was not the case when only one GAN model was used or for any of the other imputation frameworks. Nevertheless, the proposed framework with one underlying GAN model achieved better scores at missing rates of 30% and 50%. Secondly, on the Compas recidivism data, the MissForest framework achieved a better score at a 10% missing rate. Although, at higher missing rates of 50% and onward, the proposed framework achieved the best scores. Lastly, for the Adult Income data set, the TGAIN framework did not achieve the lowest overall prediction error on lower missing rates of 10% and 30%. However, similarly to the other data sets, it achieved the best scores at higher missing rates of 50% and beyond.

Thus, the proposed framework with three underlying GAN models tends to reliably achieve lower overall prediction errors at higher missing rates, even at different training set sizes. Although, at lower missing rates, TGAIN could not reliably achieve the best scores for all utilized data sets. The performance gain generally tends to be rather small when comparing the proposed framework to the other variation containing one underlying GAN model. Despite the computational load being three times as much compared to this other variation.

### 5.3.2 Results post-imputation classifier performance

| | Model Accuracy scores (in %) | | |
|---|---|---|---|
| **Framework** | German | Compas | Adult |
| TGAIN (1 model) | **71.26** | 76.0 | 80.8 |
| TGAIN (3 models) | 70.1 | 76.6 | **83.9** |
| MissForest | 68.9 | 76.2 | 79.5 |
| MICE | 70.1 | **76.8** | 82.3 |
| KNNImputer | 65.5 | 75.2 | 76.5 |
| IterativeImputer | 67.8 | 74.8 | 78.4 |

Table 5.5: Accuracy scores of the random forest model on the test set post-imputation using the different imputation frameworks, a higher accuracy score implies better overall imputations

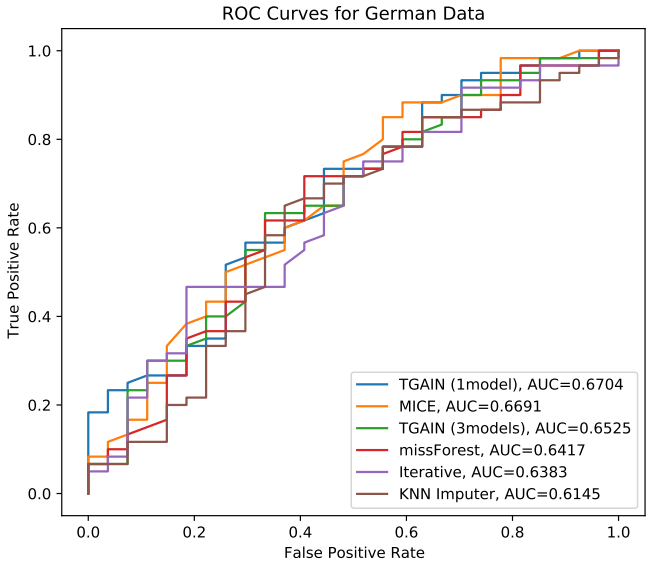


Figure 5.7: ROC-AUC scores for the different imputation frameworks trained on the German credit data set, a higher area under the curve implies a better model

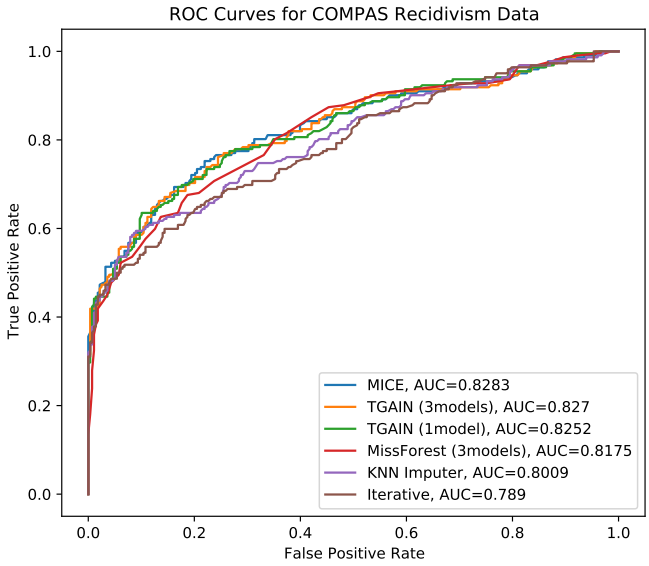


Figure 5.8: ROC-AUC scores for the different imputation frameworks trained on the COMPAS Recidivism data set, a higher area under the curve implies a better model

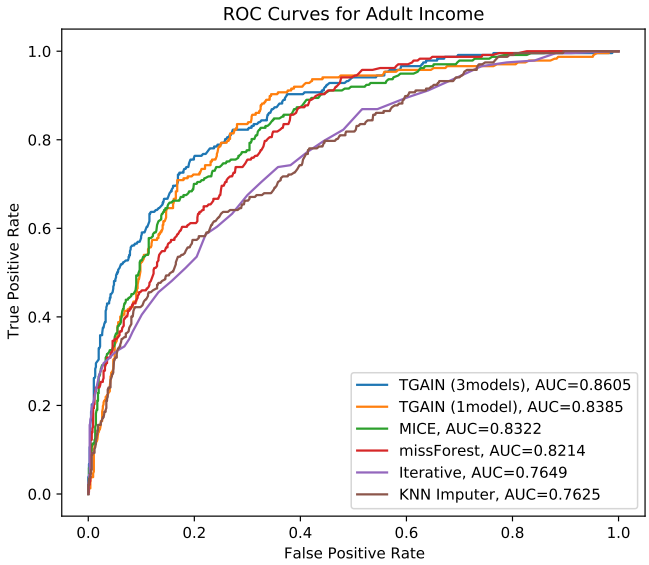


Figure 5.9: ROC-AUC scores for the different imputation frameworks trained on the Adult Income data set, a higher area under the curve implies a better model

As can be concluded from the results, the utilized imputation framework can have a significant effect on the resulting model performance. For instance, the TGAIN framework with one GAN model achieved an accuracy score of 71.26 compared to 65.5 when using KNNImputer for the German data set. This performance difference is again confirmed by these two frameworks' differences in AUC scores. Notable is that the MICE framework achieved the same accuracy score as the proposed framework with three GAN models and even achieved a slightly higher AUC score. Similarly, MICE achieved a slightly higher accuracy- and AUC score in comparison with the TGAIN framework in the Compas data set. Nevertheless, the performance difference between TGAIN, MICE, and MissForest was not significant for this data set. The highest difference in performance was visible in the adult income data set; the large number of missing values that needed to be imputed might be a possible explanation for this. The proposed framework achieved the highest accuracy score of 83.9 and an AUC score of 0.8605. Although MICE achieved a higher accuracy score than the alternative TGAIN version, it did not receive a higher AUC score.

Based on the results of this evaluation metric and the previous one, it can be concluded that the alternative TGAIN framework with one model might be better suited in many cases. To clarify, the proposed framework only reliably achieved better scores at higher missing rates, but the performance difference is not proportional to the additional computation time that comes with it. In some cases, the proposed framework could not achieve higher scores than MICE or MissForest. Nevertheless, for the situations where TGAIN achieves the best scores, it does so with a larger difference compared to the situations where MICE/MissForest achieves the best score. Moreover, TGAIN tends to outperform the KNNImputer and IterativeImputer for both evaluation metrics reliably. MissForest, on the other hand, has shown to be less consistent than the proposed framework since it achieved the second-worst performance on the German data set. Lastly, the MICE imputation framework has shown to be very effective for missing data imputation and has been more consistent in its results than MissForest.

# Chapter 6

# Conclusion and Future Work Directions

This last chapter shall provide a summary of the thesis. Afterward, the limitations of the proposed framework will be discussed, including ethical considerations that researchers might need to consider. Lastly, ideas for future works are provided with the aim of improving the proposed framework even further.

## 6.1 Summary

Missing data forms a very common problem for researchers, and incorrectly handling these missing values can degrade the quality of the results, reduce model performance, and may even lead to wrong conclusions being drawn from the data due to biased inferences. Hence, it is essential that these missing values are handled with the utmost care to prevent the problems mentioned above. In this project, a framework is proposed for missing data imputation of mixed-type tabular data using GANs. There have been many imputation models, but most fail to catch the statistical properties of the underlying data, and the imputations might distort the real data distribution. Thus, the proposed framework has been built with the idea of creating high-quality imputations that mimic the underlying statistical data properties more closely than other state-of-the-art imputation frameworks. The framework uses three independently trained GAN models, each providing a candidate imputation for a missing value. A final imputation is afterward chosen based on a majority vote. Optimally, we would have conditioned on all observed variables for a given row to impute the missing values. Nevertheless, as elaborated in chapter 4, this is often not achievable since a synthetic row can not always be provided due to the rejection-based sampling process. Therefore, if there are many conditions or conditions with a large number of distinct values, such as numerical variables, all synthetic rows will likely be rejected since no row that meets the requirements of all the observed variables could be sampled. For this reason, a solution has been implemented which aims to generate up to 50 synthetic rows conditioned only on the categorical variables. Then, using the euclidean distance metric, the top three rows with the most similar values compared to the row that needs to be imputed are found. Three candidates are chosen from the three synthetic rows, and a final imputation is likewise chosen based on a majority vote.

The proposed framework has been compared to other state-of-the-art imputation methods, using two evaluation metrics and three different data sets. First, missing values have been introduced in these data sets for a growing number of missing rates. Then, these values are imputed, and the prediction error is calculated between the imputed values and the actual observed values. However, common-used error metrics such as the root mean square error can not be used since there is a mix of continuous- and categorical variables in the data. Hence, a novel prediction error formula is presented, which can be applied to mixed-type variables and ensures that both variable types equally contribute to the overall prediction error. The proposed framework has shown to reliably outperform the baseline methods on high missing rates of 50% and beyond. In addition, the framework was competitive for lower missing rates but could not reliably achieve the best scores for all three data sets. The second evaluation metric is based on the performance of a random-forest classifier post-imputation using different imputation frameworks. Missing data have been introduced in the data set and are afterward imputed using these various imputation frameworks. Once a complete data set has been achieved, the data is used as a training set to fit the random-forest classifier. The random-forest model is then used to classify a 20% test set that was previously reserved. Accuracy- and ROC-AUC scores are calculated to determine the best model, from which afterward the best imputation model can be deduced. From the results, the conclusion can be made that the utilized imputation framework can greatly affect the model performance. Both the proposed framework and the MICE algorithm provided the best classifier models, and are therefore seen as the better imputation frameworks. In the next section, the limitations of the proposed framework shall be discussed.

## 6.2 Limitations

The proposed framework has several limitations that might pose a problem for researchers. Firstly, as elaborated in section 4.2, training the GAN models might be unstable depending on the utilized data set. It is essential that the generator and discriminator converge to each other such that one neural network does not dominate in strength over the other. If the generator overpowers the discriminator, this might ultimately result in low-quality data imputations. Likewise, if the discriminator becomes too strong too quickly, the generator might fail to learn from the discriminator. Researchers should therefore examine the discriminator- and generator loss throughout the training iterations and adjust the learning rates accordingly to ensure that the networks converge to an equilibrium. The next limitation is the possible time complexity of imputing the missing values. In figure 4.6, it becomes clear that when the number of conditions is high, the imputation time can increase tremendously. Hence, researchers need to take into account the number of missing values in their data set, the number of conditions to use, and the complexity of these conditions. The number of conditions can indirectly be effected by adjusting the distinct ratio threshold when using the proposed framework.

## 6.3 Ethical considerations

The proposed framework does not lead to ethical concerns, since missing values are imputed with already existing values in the data set. Noteworthy is that this framework aims to impute all missing values in the data set, thus it does not differentiate sensitive attributes from non-sensitive attributes. Hence, researchers may opt to remove sensitive attributes such as gender and race from their data sets before utilizing the framework. The same principle applies to specific existing sensitive values in the data set. Researchers can simply remove these instances from the data set such that they are not used for imputation at a later stage. Nevertheless, utilizing imputation frameworks in general might not be a suitable solution for all scenarios where missing values need to be handled. Suppose the post-imputed data

set is used to raise the performance of a classification model, like in the second evaluation method. In that case, it might be important to consider the classification model's impact on society. To illustrate, take a classification model used to evaluate whether people qualify for a loan, and an individual is denied that loan due to the values that have been imputed. This may be problematic because an individual might be denied the loan as an indirect result of the imputation framework, since the loan might have been accepted if the missing values were imputed with other values. Therefore, a more vigilant approach might be necessary for situations where the end-use case of the data set is to build a model which may have a high impact on people's lives. One appropriate method, which may be feasible in certain cases, is to simply acquire additional observed data for instances where the model's output is highly affected due to the imputations.

## 6.4 Future work

Several directions can be investigated to improve the proposed imputation framework further. As mentioned in theoretical analysis, the assumption is made that variables are independent of each other. Take, for example, the two variables 'start_date' and 'end_date', where only one has been observed and the other needs to be imputed. In the current proposed framework, the situation can occur where the missing value is imputed with a date that lies before the start date or after the end date. Since these imputations would be insensible, the researchers could improve the overall quality of the imputations by implementing a method that automatically finds sets of arguments for each variable, which the imputed value needs to satisfy before being accepted. Nevertheless, this shall likely increase the computational load of the proposed framework even further since more samples might be rejected. Hence, another direction would be to reduce the computational complexity of the framework. More efficient imputation algorithms could be implemented, and other time-reducing methods such as investigating the effect of utilizing fewer layers in the neural networks on the results could be assessed. Lastly, only implicit missing values have been imputed using the proposed framework in this research. However, researchers could focus on automatically identifying explicit missing values (disguised under a fake value) and impute those too. Namely, it might be interesting to determine whether imputing explicit missing values in the data set could lead to a better classification model than not imputing these disguised missing values.

# Bibliography


[1] M. Arjovsky and L. Bottou. "Towards Principled Methods for Training Generative Adversarial Networks". In: ((2017)). doi: 10.48550/ARXIV.1701.04862.

[2] S.E Awan, M. Bennamoun, F. Sohel, F. Sanfilippo, and G. Dwivedi. "Imputation of missing data with class imbalance using conditional generative adversarial networks". In: *Neurocomputing* 453 ((2021)). doi: https://doi.org/10.1016/j.neucom.2021.04.010, pages 164–171. ISSN: 0925-2312.

[3] S. Buuren and C. Groothuis-Oudshoorn. "MICE: Multivariate Imputation by Chained Equations in R". In: *Journal of Statistical Software* 45 ((2011)). doi: https://doi.org/10.18637/jss.v045.i03.

[4] S. Cheng-Xian Li, B. Jiang, and B.M. Marlin. "MisGAN: Learning from Incomplete Data with Generative Adversarial Networks". In: ((2019)). doi: https://doi.org/10.48550/ARXIV.1902.09599.

[5] W. Dong, D. Fong, J. Yoon, E. Wan, L. Bedford, E. Tang, and C. Lam. "Generative adversarial networks for imputing missing data for big data clinical research". In: *BMC Medical Research Methodology* 21 ((2021)). doi: https://doi.org/10.1186/s12874-021-01272-3.

[6] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. "Generative Adversarial Nets". In: 27 ((2014)). https://doi.org/10.48550/arXiv.1406.2661.

[7] R.H.H Groenwold and O.M Dekkers. "Missing data: the impact of what is not there". In: *European Journal of Endocrinology* 183 ((2020)). doi: https://doi.org/10.1530/EJE-20-0732, pages 7–9. ISSN: 0804-4643.

[8] R. Lall. "How Multiple Imputation Makes a Difference". In: *Political Analysis* 24 ((2017)). doi: https://doi.org/10.1093/pan/mpw020, pages 414–433.

[9] D.T Neves, M.G Naik, and A. Proença. "SGAIN, WSGAIN-CP and WSGAIN-GP: Novel GAN Methods for Missing Data Imputation". In: *Computational Science – ICCS 2021*. doi: https://doi.org/10.1007/978-3-030-77961-0_10. Springer International Publishing, (2021), pages 98–113.

[10] A. Qahtan, A. Elmagarmid, M. Ouzzani, and N. Tang. "FAHES: Detecting Disguised Missing Values". In: *IEEE 34th International Conference on Data Engineering* ((2018)). doi: https://doi.org/10.1109/ICDE.2018.00188, pages 1609–1612.

[11] T. Redman. "The Impact of Poor Data Quality on the Typical Enterprise." In: *Communications of the ACM* 41 ((1998)). doi: https://doi.org/10.1145/269012.269025, pages 79–82.

[12] J.L. Schafer and J.W. Graham. “Missing Data: Our View of the State of the Art”. In: *Psychological Methods* ((2002)). doi: https://doi.org/10.1037/1082-989X.7.2.147, pages 147–177.

[13] L. Sixt, B. Wild, and T. Landgraf. “RenderGAN: Generating Realistic Labeled Data”. In: ((2016)). doi: https://doi.org/10.48550/ARXIV.1611.01331.

[14] D. Stekhoven and P. Bühlmann. “MissForest - Non-parametric missing value imputation for mixed-type data”. In: *Bioinformatics (Oxford, England)* 28 ((2012)). doi: https://doi.org/10.1093/bioinformatics/btr597, pages 112–118.

[15] “Volume of data/information created, captured, copied, and consumed worldwide from 2010 to 2025”. In: 32 ((2022)). url: https://www.statista.com/statistics/871513/worldwide-data-created/.

[16] L. Xu, M. Skoularidou, A. Cuesta-Infante, and K. Veeramachaneni. “Modeling Tabular data using Conditional GAN”. In: 32 ((2019)). doi: https://doi.org/10.48550/arXiv.1907.00503, pages 7335–7345.

[17] J. Yoon, J. Jordon, and M. van der Schaar. “GAIN: Missing Data Imputation using Generative Adversarial Nets”. In: Proceedings of Machine Learning Research 80 ((2018)). https://proceedings.mlr.press/v80/yoon18a.html, pages 5689–5698.

[18] S. Yoon and S. Sull. “GAMIN: Generative Adversarial Multiple Imputation Network for Highly Missing Data”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. doi: https://doi.org/10.1109/CVPR42600.2020.00848. (2020), pages 8453–8461.

[19] J. Zhu, T. Park, P. Isola, and A. Efros. “Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks”. In: *CoRR* (2017). doi: https://doi.org/10.48550/arXiv.1703.10593.