



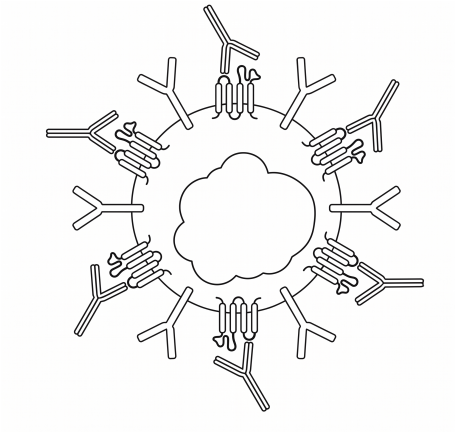


Bachelor Thesis

Synthetic approach to data augmentation for Nodular Lymphocyte Predominant Hodgkin Lymphoma

Carlotta de' Castiglioni - mat. 512944
Artificial Intelligence

Academic Year 2024/2025



Bachelor in Artificial Intelligence
Pavia 2026

Bachelor in Artificial Intelligence







Carlotta de' Castiglioni - mat. 512944

Synthetic approach to data augmentation for Nodular Lymphocyte Predominant Hodgkin Lymphoma

Supervisor: **Prof.ssa Arianna Dagliati**

© Carlotta de' Castiglioni - mat. 512944, 2026

Abstract

Keywords

Nodular Lymphocyte Predominant Hodgkin Lymphoma, Synthetic data, Generative Adversarial Network, Diffusion Model

There is no current Nodular Lymphocyte Predominant Hodgkin Lymphoma standard treatment guideline; however, the de facto treatment for these patients is the Rituximab antibody. Clinicians are thus interested in analyzing the prognostic efficacy of Rituximab; however, the major problem in attempting to study this phenomenon is data scarcity. This is partially due to the rarity of the disease and partially because the majority of subjects are currently treated, thus making it impossible to find a control arm to demonstrate the treatment effect or, at least, to capture the relationship between the Rituximab effect and the patient's conditions.

The research question of this thesis is: can this problem be mitigated with the use of synthetically generated data? A positive result could be a relevant step forward in the care of patients suffering from such a rare disease.

Starting from a real small dataset from the San Matteo Hospital in Pavia, we implemented two different methods for synthetic generation, a Generative Adversarial Network and a Tabular Diffusion Model, with the goal of recreating an entirely new tabular dataset. The requisites for the new dataset are: resemble the real dataset in order not to lose utility in the Rituximab study, and preserve privacy, guaranteeing that the original dataset cannot be reconstructed starting from the synthetic one. While the two implemented models reached suboptimal performances in terms of synthetic data utility and resemblance, this thesis demonstrates the feasibility of this approach and is the starting point for future works and advancements.

Contents

1 Introduction **3**
- 1.1 Problem Statement . . . 3
- 1.2 Outline of the project . . . 5

2 Dataset and preprocessing **6**
- 2.1 Dataset . . . 6
- 2.2 Preprocessing . . . 10
  - 2.2.1 Columns removal . . . 10
  - 2.2.2 Renaming and converting . . . 11
  - 2.2.3 Imputation . . . 12
  - 2.2.4 Right Censoring . . . 12
- 2.3 Final dataset . . . 13

3 Methods and tools **17**
- 3.1 Generative Models for synthetic data . . . 19
  - 3.1.1 GAN . . . 21
  - 3.1.2 Diffusion Model . . . 23
- 3.2 Evaluation methods for synthetic data . . . 25
- 3.3 SynthRO . . . 26
  - 3.3.1 Benchmarking . . . 28
- 3.4 Survival Analysis . . . 28

4 Results **30**
- 4.1 Analysis of the original dataset . . . 31
- 4.2 Analysis of GAN results . . . 32
- 4.3 Analysis of TabDDPM results . . . 34
- 4.4 Final score . . . 38

5 Discussion and conclusion **39**

References **41**

6 Appendix **44**
  - 6.0.1 GANs via pygan . . . 44
  - 6.0.2 Diffusion Models via Pytorch . . . 45

Chapter 1
Introduction

This chapter is going to describe the reasons for this project, identifying the needs, locating the work in current literature and describing the approaches implemented to address the case.

1.1 Problem Statement

Nodular lymphocyte-predominant Hodgkin lymphoma (NLPHL) is a rare B-cell neoplasm, accounting for approximately 5% of all Hodgkin lymphoma cases [14]. The majority of patients are male, with early-stage disease, and are diagnosed in the fourth decade of life. [4]

Usually, NLPHL has a long survival expectancy but a tendency for late and multiple relapses. Also, a subset of patients undergoes the transformation to an aggressive non-Hodgkin lymphoma. [14] The number of deaths due to therapy late effect (or deaths non-related to the lymphoma) outweigh the number of lymphoma-specific deaths. [4]

Given the rarity of this disease, very few evidences are provided and no precise guidelines for patient treatment have been published yet. Historically, NLPHL has been treated as a classic Hodgkin lymphoma. However, in most cases, NLPHL will follow a more indolent course than classical HL, with better overall survival. [7]

Usually, radiotherapy is recommended for a favorable stage, while for other stages a Chemotherapeutic-backbone strategy is suggested, with a standard more similar to classic HL.

The positivity of NLPHL cells to CD20 surface antigen has opened the door for combination therapy with anti-CD20 monoclonal antibody Rituximab that was first tested in NLPHL patients in 2007 [30].

During the following 10 years it was tested on sixteen consecutive newly diagnosed adult patients. At the end of the treatment, the effects on the group were compared to an historical cohort of other 12 patients treated with other types of treatment that were commonly used for the lymphoma. After a 7-year follow-up, the progression-free survival of patients with Rituximab was 100%, compared to 66% for the other group [30]. This result can be seen in Figure 1.1. Short and long-term side effects (including grade 3-4 neutropenia and long-term non-hematological toxicities) were more represented in the historical group, rather than the Rituximab one [30]. Additionally, more toxicities were found in the historical group compared to the Rituximab group. The results confirmed the expected efficacy of Rituximab on NLPHL patients with a positive risk-benefit ratio.

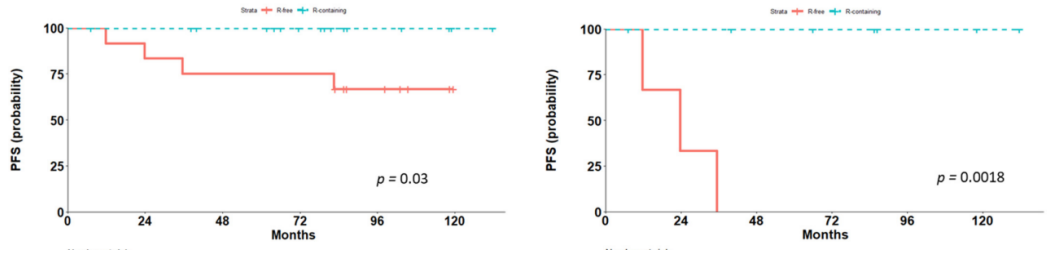


Figure 1.1: Progression-free survival for patients treated with Rituximab (in blue) and patient not treated with Rituximab (in red). This image was taken from the paper [30] and it refers to their conducted study. For more information, refer to the original paper.

Nowadays, the type of treatment implemented is based on the stage of the disease, and it can include or not radiation therapy and chemotherapy, and Rituximab immunotherapy [30]. A study [30] showed that in patients with early stage disease, Rituximab alone is less effective than radiotherapy or chemiotherapy. In particular cases, considering the condition of the patient and their risk factors, other combinations of treatments were more favorable than Rituximab. Anyways, Rituximab was well tolerated by all the patients in the study and no severe side-effects could be observed.

The situation is more complex in the late stages of the disease, where the optimal combination of treatments is harder to identify. A common idea is that Rituximab by itself is less effective when the diagnosis of the disease is made at a late stage; in this case, a combination of immuno and chemotherapy is preferred and widely used nowadays. [30]

However, all these results were obtained from a small number of patients, due to the rarity of the disease, so no definitive conclusion can be drawn. The scarcity of data is the main limitation in studying the efficacy of Rituximab on NLPHL patients.

Data availability, and data scarcity, are some of the main challenges the research world is facing. In the context of rare diseases like NLPHL, this is clear.

Another challenge the research is facing and that applies to this case is privacy concerns. Sensitive data cannot be shared carefree: clinical data are collected from real patients and they must be protected. For this reason, retrieving it can be even harder.

Ultimately, ethics here plays a fundamental role. Given that it is known the positive outcome patients have when Rituximab is added to their treatment, doctors cannot decide not to administer Rituximab for the sole purpose of collecting data about non-treated patients.

Considering now the research question that clinicians are interested it, i.e. for which patient profile Rituximab is more effective, and under which aspects, and their major problem, i.e. scarcity of data, we propose a new research question: can the first question be answered with the implementation of synthetic data?

During the last decades we assisted to an incredible merge between medical practice and technologies. This new discipline, Health Informatics, is a fundamental step for improving and accelerating medical research, and this progress is based on an always more reliable analysis of clinical and biomedical data. In particular, trying to retrieve the real pattern of data, with the use of complex algorithms. This is particularly useful in those studies that track the time between a study baseline and an event of interest (e.g., disease progression or treatment response).

The goal of our project is to assess the problems of data scarcity and privacy and ethics concerns, generating a synthetic dataset derived from an original, real-world dataset of NLPHL patients. This dataset is supposed to be good enough to be implemented in helping medical researchers to understand the relationship between Rituximab and the dis-

ease, and good enough to resemble the original dataset, but also diverse enough to not be able to retrieve the original patients from the synthetic dataset.

We had the opportunity to face the problem directly with the staff of the hematology department of San Matteo Hospital in Pavia. They have a collection of data about patients with NLPHL, small enough to create major limitations when studying the Rituximab effect on their patients.

The primary goal of this project is to identify a generating tool to compare the progression free survival between two treatment cohorts (Rituximab containing and Rituximab free), being sure that the generated dataset resembles the original one and that original patients cannot be retrieved giving just the synthetic dataset.

1.2 Outline of the project

The study is organized as shown in Figure 1.2. We started analyzing the original dataset provided by the hospital and performed an accurate data preprocessing (Chapter 2), then the clean dataset was given as input to synthetic generation methods (Chapter 3) producing a new bigger dataset different from the original one. This resulting dataset underwent an evaluation process to check its quality (Chapter 4).

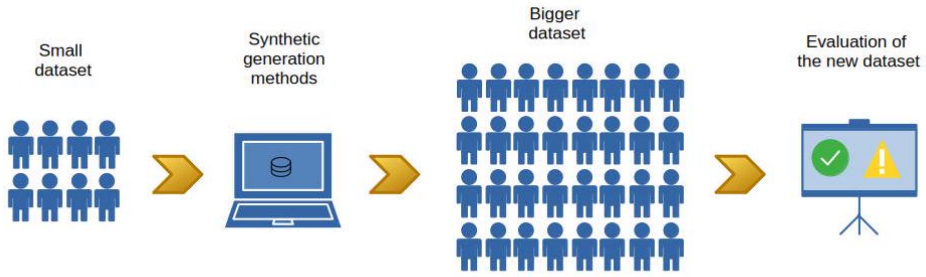


Figure 1.2: Structure of the study

Chapter 2
Dataset and preprocessing

2.1 Dataset

The starting point of this project is a real-world, anonymized, patient-level dataset related to Nodular Lymphocyte Predominant Hodgkin Lymphoma (NLPHL), provided by the IRCCS Policlinico San Matteo in Pavia. The data was provided in a structured, tabular format, constituting a patient-feature matrix where each row represented a unique patient and each column contained a specific clinical feature.

The group of patients was defined as an electronic phenotype of NLPHL, meaning patients were included based on the manifestation of the disease.

This group of individuals with NLPHL was identified at a baseline point in time and observed over a period. A control group of individuals that were not treated was also present in the original data repository; however, the scope of this project is focused exclusively on the treated cohort. In particular, we were interested in evaluating the effect of treating a patient with a specific treatment: Rituximab.

The original dataset presented several challenges common to retrospective clinical data. For this reason, the raw data required extensive preprocessing. Inconsistencies included non standardized categorical values, missing entries, redundant columns, and potential data entry errors that necessitated a rigorous data cleaning.

Another aspect to consider was the Loss to Follow-up: the unequal and incomplete follow-up across the cohort. In theory, follow-up procedures should be applied equally to ensure validity, but in practice loss of contact with subjects was present, leading to temporal data heterogeneity.

From a statistical point of view, we assumed that the patient samples were independent, non replicated.

Before addressing synthetic data generation, a comprehensive inspection of all the features was conducted. This step was critical to understand the underlying structure in the source data, which the synthetic dataset must accurately reflect.

All features in the original dataset that we received are described in Table 2.1

Table 2.1: Features of the original dataset

Feature name	Description
PZ	An indicative patient ID to order the data
CENTRO	Referring to the center where the patient is from. That could be: Bergamo, Brescia, Firenze, Guido Gini Hospital, Humanitas, INT Milano, MI Niguarda Hospital, Monza, Napoli, Novara, Padova, Pavia, Perugia, Reggio, Roma, Siena, Torino, Torino Univ, Varese
NumeroschedaesPAV001	An alphanumeric value corresponding to the center and an internal number (for example: PAV001)
sessoM0F1	A letter indicating if the patient was female or male
DATAESORDIO	The date when the disease is first diagnosted
DATAINIZIOTRP, DATAFINETRP	Respectively the date from when the treatment started and when it ended. If it didn't end, the last date was a censoring on the November 15th, 2019
ULTIMOFUP_x_analisi	In most of the cases it corresponds to November 15th, 2019. Alternatively, the last follow up date
data_pfs	Date indicating the date when the disease progressed (if it didn't progress: November, 15th 2019)
ETAALLESORDIO	The age of the patient when the disease was first diagnosed
eta45	Indicating if the patients is either younger or older-or-equal than 45 years. It's a boolean column. This suggests from the beginning that that age is a relevant threshold for studying the course of the disease
STADIAZIONE_PET_TC	An integer indicating in which stadium the lymphoma is (1, 2, 3, 4)
stadio_cat2	Indicating if the stadium is 1 or >= 2
stadio_cat3	Indicating if the stadium is in 1,2 or in 3,4
SINTOMIB	Lymphoproliferative diseases can either be symptomatic or asymptomatic. Symptoms are called "B symptoms", and they are: profuse night sweats, low-grade fever serotonin, weight loss not related to diet (10% of weight in 6 months). The presence of B symptoms is indicated in this column as presence or absence (binary values)
WBC103mmc	Indicating the total number of white blood cells. The unit of measure is $10^3 mm^3$. Here, *mmc* means that they are expressed as absolute value. Values go from 2.54 to 9.91
NEUT	It refers to the quantity of neutrophils (white blood cells), expressed as percentage. Values go from 64.4 to 94
LINFO	It refers to the quantity of white blood cells, expressed as percentage. Values go from 2 to 56

Table 2.1: Descriptive Statistics (continued)

Feature name	Description
linfo8	It indicates if LINFO column is either smaller or greater-or-equal than 8
MONO	It refers to the quantity or monocytes (white blood cells), expressed as percentage. Values go from 8.3 to 9.9
HGBgdl	It indicates the quantity of haemoglobin, express in g/dl. Values go from 9 to 17.7
hb105	It indicates if HGBgdl is either smaller or greater-or-equal than 10.5
PLT103mmc	It indicates blood platelets, expressed in $10^3 mm^3$ (absolute value). Values go from 63 to 533
VESmmh	It indicates the erythrocyte sedimentation rate (parameter for inflammations), measure in mm/h. Values go from 1 to 90
VEScat	It indicates if VES is either smaller than 30, between 30 and 50, or greater than 50
ESR_VES_aumentata	VES is said to be augmented if it's greater than 30 mm/h. It' a binary column with values 0 and 1
LDHelevato	LDH is an enzyme called lactate dehydrogenase. It is released from cells when proliferation increases (typical in hematological tumors). The threshold value is 220 internation units over liter. This is binary column with value 1 if it's greater than the threshold, 0 otherwise.
beta_elevato	It stands for beta-2-microglobulin. This protein is part of the lymphocyte receptors, and it usually increases in high-turnover lymphoproliferative disorders. Given the standard threshold, the value is 1 if it's greater, 0 otherwise.
ALBgdl	Albumin is a blood protein measured in g/dl. It takes values from 2.7 to 5.74
alb4	The threshold for albumin is 4 g/dl. This column indicates if albumin is either smaller or greater-or-equal than 4
MILZA0no1si3NA	The spleen is a an organ that acts as a big lymphonode. This means that, just as lymphonode increases their dimension, also spleen increases its dimension. Takes value 0 if the spleen dimension was normal, 1 if increased.
LINFADDOM	Hodgkin lymphomas usually starts involving supra-diaphragmatic lymphones: neck, armpit, and chest lymphonodes. Takes value 1 if this is the case, 0 otherwise.
Areelinfonodalisovradinteres	It refers to the number of supra-diaphragmatic lymphones involved. Values range from 0 to 5
areelinfosovradiaframma3	This is a redundunt column of the previous one

Table 2.1: Descriptive Statistics (continued)

Feature name	Description
BOMposneg	This means bone marrow biopsy. Sometimes the disease can be found also in the bone marrow. For this reason, a biopsy of the bone narrow is a useful method to investigate. This column takes value *pos* if the disease if found also in the bone marrow, *neg* otherwise
Numsediextranodali	It indicates the number of extra nodal sites. This is because lymphomas can involve any organ. Values range from 0 to 4
NsediExtra_noBOM	It indicates the number of extra nodal sites, without considering BOM. Values range from 0 to 4
FEGATO	It indicates if the liver is one of the extra nodal sites. Takes value 1 if it is, 0 otherwise.
OSSO_VERTEBRE	It indicates if vertebrae are one of the extra nodal sites. In this dataset, no patient had this feature, reasons for why the only value was 0
bulky	A bulky mass is so defined when its dimension is greater than 7 centimeters. Takes value 1 if it's greater, 0 otherwise
TERAPIA SI VS NO	Revealing if the patients is actually undergoing a therapy or if they are just under "observation"
Trattamento_Descrizione	A numerical value corresponding to the type of treatment
Trattamento	The name of the treatment given to the patient
RISPOSTA	Final evaluation after the treatment, performed with TAC and PET, we define *NR* (no-response) if the disease didn't substantially change (change below 25%), *SD* (stable disease) if the change is between 25% and 50%, *PR* (partial response) if the change is greater then 50%, *RC* (complete response) if the change is 100%. *PD* (progressive disease) indicates that the disease has worsened despite treatment.
RISP_cat	This is just a repetion of the column *RISPOSTA*
ORR	This is the overall response. We consider it 1 if the response is either *PR* or *CR* (partial or complete response), while it's 0 if the response is either *SD*, *NR*, *PD*.
STATOMALATTIA	This indicates if the disease is stable or progressing after the final evalutation. This can be *RC*, *RP*, *SD*.
motivo_decesso_cat	If the patient died, here is indicated the cause. Example: mallatia dei legionari (Polmonite). Otherwise it's a NaN value
toxpolmonaresino	In chemotherapy for Hodgkin lymphomas a medicine called bleomycin is implemented, but it's usually associated with a polmonary toxicity. This columns indicates which patients suffered from polmonary toxicity

Table 2.1: Descriptive Statistics (continued)

Feature name	Description
evento_os	Indicating if there was an event that caused the death of the patient (binary 0-1)
evento_pfs	Indicating if the disease progressed (binary 0-1)
TIME (GG)	The number of days passed between the beginning of the treatment and evento_pfs

2.2 Preprocessing

The dataset that we received contained many inaccuracies that had to be fixed. This section is going to explain the detailed analysis of the work.

The data cleaning procedure was made with the help of a simple script in R, but also with the help of a constant manual checking. Since the columns were all very different from each other, we had to make sure of all the steps that we were doing. We needed to keep track of the process and be able to decide how to intervene in every step.

The first action was to remove a single row were the days were inserted wrong and the end-of-treatment resulted before the start-of-treatment, and to remove a single row where the Time (GG) value was a too far outlier, probably due to an inserting error.

In the next subsections, we will make a selection of the columns in the dataset, rename and convert some values, impute missing values and perform a right censoring.

2.2.1 Columns removal

A common problem in Machine Learning methods is the so called curse of dimensionality [9]. This happens when models are supposed to learn from a wide range of features containing wide range of values. A big number of features does not automatically imply that the model will have a better performance in learning the features. Most of time, models may lower the quality of the results whenever the number of variables increases after a certain amount [9].
To prevent this, a selection of the available features is done.

We removed the columns that we were not going to use in any case, because they were not relevant for the analysis. These columns were: *PZ, CENTRO, NumeroschedaesPAV001*, because we were not intrerested in the origin of the patient, and *motivo_decesso_cat*, since the cause of death was not relevant in our study.
Additionally, *DATAESORDIO, DATAINIZIOTRP, DATAFINETRP* were removed because they were already summarized in *TIME (GG)*.

Then we removed all the columns that were a dependency from another one. For example, the column *eta45* was a dependency of the column *ETAALLESORDIO*, since *ETAALLESORDIO* contained a numerical value indicating the age of the patient and *eta45* was a binary feature indicating 0 if the patient was younger than 45 years old, and 1 otherwise. The removed columns were: *stadio_cat2, stadio_cat3, linfo8, hb105, RISPOSTA, Trattamento_Descrizione, evento_os* (all the events in *event_os* were included in *evento_pfs*, that was our main object of interest), *data_pfs* (already included in *Time (GG)*).

After this screening, we considered the fact that our dataset was actually very small for synthetic data generation, so we needed to remove all the columns that were not strictly fundamental. Increasing the number of features plays against the stability of the model, especially when the sample size is small; that's why we needed to reduce the number of input features to the minimum [2].

Those columns were: *VESmmh, VEScat, alb4, Areelinfonodalisovradinteres, areelinfosovradiaframma3, Numsediextranodali, NsediExtra_noBOM, WBC103mmc, NEUT, LINFO, ULTIMOFUP_x_analisi, RISP_cat, ORR, STATOMALATTIA, MONO, Durata_Terapia (GG).*

The decision of which columns to remove was made with the help of the clinical staff.

Since our study focuses on the effect of Rituximab against a treatment without the use of Rituximab (and not against no treatment at all), we did not need the patients who were included in the study but were just under observation. So, we removed those rows. They were 14 out of 308 total patients.

We also took a look at the NaN values. Just by looking at the dataset, we could see that we had many of them. We printed a list of all the columns with NaN values and the proportion of them. Always thinking about the fact that we had to simplify the dataset as much as possible, we removed the columns that had more than 20% of NaN values. Those were: *beta_elevato, ESR_VES_aumentata, ALBgdl, MONO*.

An important step was to look at the balance of the columns. We could see that some binary columns had almost just one value, with a very small appearance of the sibling value. This can result in a very difficult training for generating models, together with the small size of the dataset.

In particular, *SINTOMIB* had a proportion of 90.8% vs 9.2%, *BOMposneg* has 98.4% vs 1.6%, *FEGATO* has 97.6% vs 2.4%, *OSSO_VERTEBRE* contained just one possible value, *bulky* had 95.2% vs 4.8%, *RISP_CAT* had three values with proportions 92.9% vs 4.4% vs 2.7%, *ORR* had 97.3% vs 2.7%, *STATOMALATTIA* had 97.6% vs 2.1% vs 0.3%, *toxpolmonaresino* had 94.3% vs 5.7%. All these columns that we just named were removed from the dataset.

2.2.2 Renaming and converting

Some columns were full of more than one value meaning the same thing. For example, in a single columns we could observe values such as *no, NO, si, SI, sí*. That was clearly due to an oversight.

Since we were working with this renaming process, we also converted all the alphabetic values inside categorical columns into numerical values. For example, the *sessoM0F1* columns containing values *M* and *F*, indicating *maschio* and *femmina*, were converted in a column containing *0* and *1*. This step was applied to all the columns of this kind and was a necessary step for the encoding in the model that we will use to generate synthetic samples.

The *Trattamento* column contained many different values, referring to the type of treatment and the use or Rituximab. There were in total six different types of treatment, and two options for Rituximab, that was either the use of it or not. These two characteristics are combined to obtain 12 possible values. For example, one element could be "R-CHOP" indicating that the patient was treated both with Rituximab and Chop therapy, or "ABDV" indicating that the patient was treated with chemotherapy alone.

We decided to create an additional column referring to the use of Rituximab. This is a binary column called *Rituximab* that takes value 0 if Rituximab was not given to the patient, and value 1 if it was involved in the treatment. The original *Trattamento* column could now be described just with 6 possible values. In particular, the possible values were:

- No additional treatment: since we removed the patients with no treatment at all, the patients with value 0 in this column will have value 1 in the Retuximab column.
- ABDV: the patient was treated with chemiotherapy
- RADIO: the patient was treated with radiotherapy
- ABVD and RADIO: the patient was treated with both chemiotherapy and radiotherapy
- CHOP: the patient was treated with CHOP therapy
- CHOP and RADIO: the patient was treated with both CHOP therapy and radiotherapy

  Also here, these six options were converted to a range of integers from 0 to 5.

Another renaming step was made in the name of the columns, just for more clarity. *sessoM0F1* was renamed *sesso*, *MILZA0no1si3NA* was renamed *milza*, *STADIAZIONE_PET_TC* was renamed *STADIAZIONE*.

2.2.3 Imputation

At this point we addressed the problem of missing data. Here, missing data referred to data that should have existed but were not recorded: a classic Missing At Random (MAR) scenario [16], very common in clinical databases.

A common strategy to solve this problem is implementing a multiple imputation (MI) [20], which addresses this by creating several complete datasets. It analyzes all of them and then recombines them into a new single dataset without missing data.

We decided to use the Multivariate Imputation by Chained Equation (MICE) [33] package in R, because it has a powerful and accurate MI method. As a classic MI, MICE package is able to deal both with numerical and categorical data, and it creates multiple plausible dataset with imputed values, iteratively modelling each missing values as a function of the other features. This datasets are then analyzed separately. MICE method guarantees that the result is going to be a valid statistical inference. As default parametric imputation Predictive Mean Matching (PMM) [21] was used for numerical features. The only columns that contained NaN values, after the data cleaning steps, were *HGBgdl*, *PLT103mmc* and *LDHelevato*: all of them being numerical columns. So, we didn't need to deal with categorical columns imputation. We used standard $\mathtt{iterations} = 5$.

The specific procedure inside MICE can be seen in Figure 2.1.

2.2.4 Right Censoring

Since we will be interested in a performing survival analysis, we decided to apply a right-censoring [23] at five years (1825 days). Right censoring is a procedure for which all the patients that never had an event (or had an event after a fixed time) were censored at *time=fixed time*; patients with loss of follow-up were censored at that last known date; patients with an event before the fixed time had their own time. In our case, the fixed time is 1825 days.

The standard assumption is that this is a type of non-informative censoring: a patient's censoring does not predict their future event risk.

Initially, this thorough censoring was applied to the starting dataset. After working on synthetic generation, we realized that it was more reasonable to let the model generate

[1] `https://cran.r-project.org/`

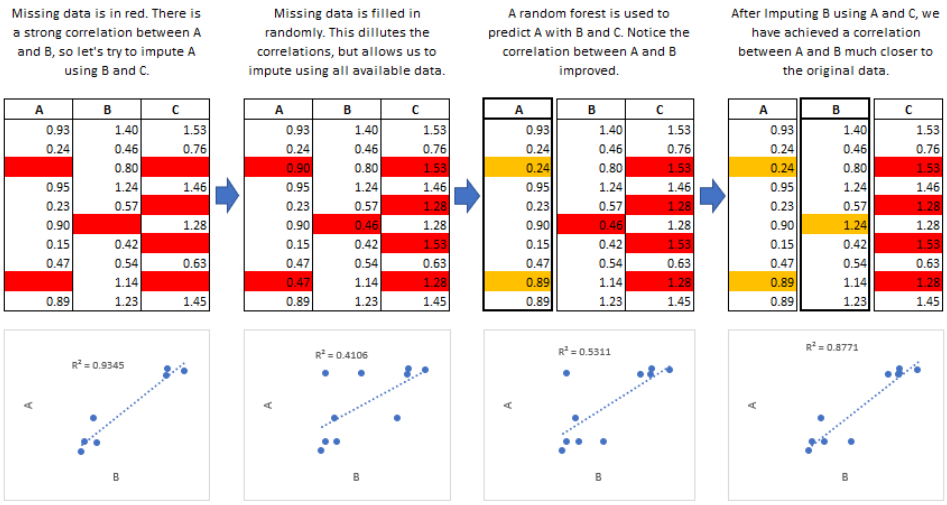


Figure 2.1: MICE procedure. This image was taken from the original source of MICE package [1]

data without censoring, and then apply the censoring. So, we will be doing this after the synthetic generation. This will be explained in the methods section (Chapter 3).

2.3 Final dataset

After the steps we described in the previous section, except for the right censoring, the data was ready to be fed into the generating models.

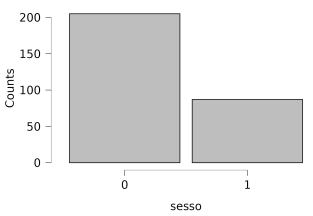
The dataset contains data about 293 patients, each patient is represented by the features described in Table 2.2.

Some plots representing the distributions are displayed in Figure 2.2.

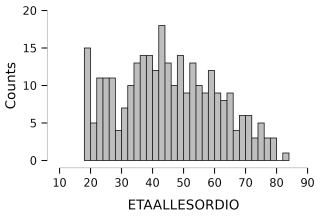
Additionally, Figure 2.3 shows the correlation matrix between the columns. The major correlation is shown between the sex of the patient of their level of hemoglobin. This is reasonable, since males have higher hemoglobin levels in general. Other correlations are between the stadium of the disease and the dimension of the spleen, and between the stadium and supra-diaphragmatic lymphones involvement. Also, LDH levels are related to the splenic involvement.

		Overall	No-Rituximab	Rituximab	p-value
n		292	150	142	
sesso (%)	0	205 (70.2)	109 (72.7)	96 (67.6)	0.414
	1	87 (29.8)	41 (27.3)	46 (32.4)	
ETAALLESORDIO (mean (SD))		45.22 (15.78)	41.61 (14.82)	49.02 (15.91)	<0.001
STADIAZIONE (%)	1	80 (27.4)	61 (40.7)	19 (13.4)	<0.001
	2	122 (41.8)	61 (40.7)	61 (43.0)	
	3	57 (19.5)	21 (14.0)	36 (25.4)	
	4	33 (11.3)	7 (4.7)	26 (18.3)	
HGBgdl (mean (SD))		14.21 (1.61)	14.32 (1.71)	14.08 (1.48)	0.212
PLT103mmc (mean (SD))		243.36 (82.82)	250.73 (81.78)	235.58 (83.49)	0.118
LDHelevato (%)	0	255 (87.3)	140 (93.3)	115 (81.0)	0.003
	1	37 (12.7)	10 (6.7)	27 (19.0)	
milza (%)	0	258 (88.4)	141 (94.0)	117 (82.4)	0.004
	1	34 (11.6)	9 (6.0)	25 (17.6)	
LINFADDOM (%)	0	159 (54.5)	96 (64.0)	63 (44.4)	0.001
	1	133 (45.5)	54 (36.0)	79 (55.6)	
Trattamento (%)	0	10 (3.4)	0 (0.0)	10 (7.0)	<0.001
	1	80 (27.4)	36 (24.0)	44 (31.0)	
	2	111 (38.0)	76 (50.7)	35 (24.6)	
	3	40 (13.7)	1 (0.7)	39 (27.5)	
	4	9 (3.1)	0 (0.0)	9 (6.3)	
	5	42 (14.4)	37 (24.7)	5 (3.5)	
evento_pfs (%)	0	234 (80.1)	112 (74.7)	122 (85.9)	0.024
	1	58 (19.9)	38 (25.3)	20 (14.1)	
TIME..GG. (mean (SD))		2768.57 (1822.33)	3369.27 (2044.69)	2134.04 (1283.14)	<0.001

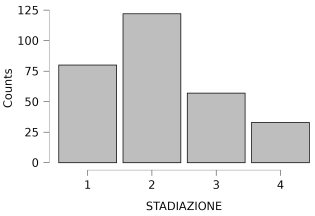
Table 2.2: Analysis of the distribution of the features of the original dataset.



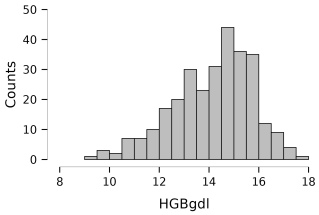
(a) sesso column distribution



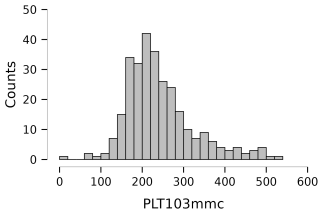
(b) ETAALLESORDIO column distribution



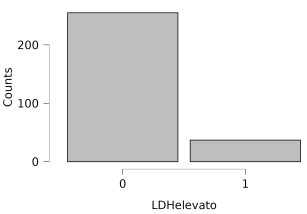
(c) STADIAZIONE column distribution



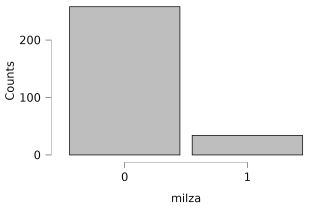
(d) HGBgdl column distribution



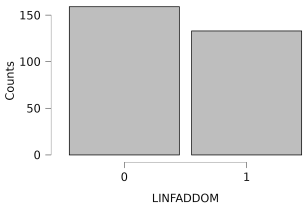
(e) PLT103mmc column distribution



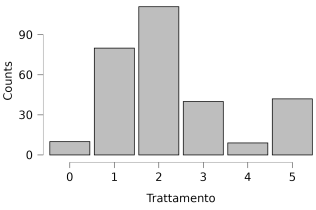
(f) LDHelevato column distribution



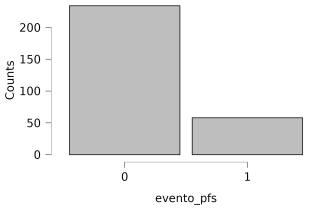
(g) milza column distribution



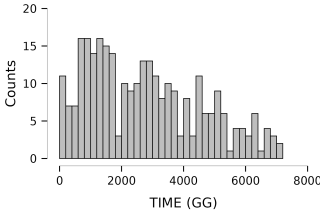
(h) LINFADDOM column distribution



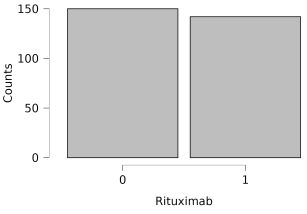
(i) TRATTAMENTO column distribution



(j) evento_pfs column distribution



(k) TIME(GG) column distribution



(l) Rituximab column distribution

Figure 2.2: Dataset column distribution

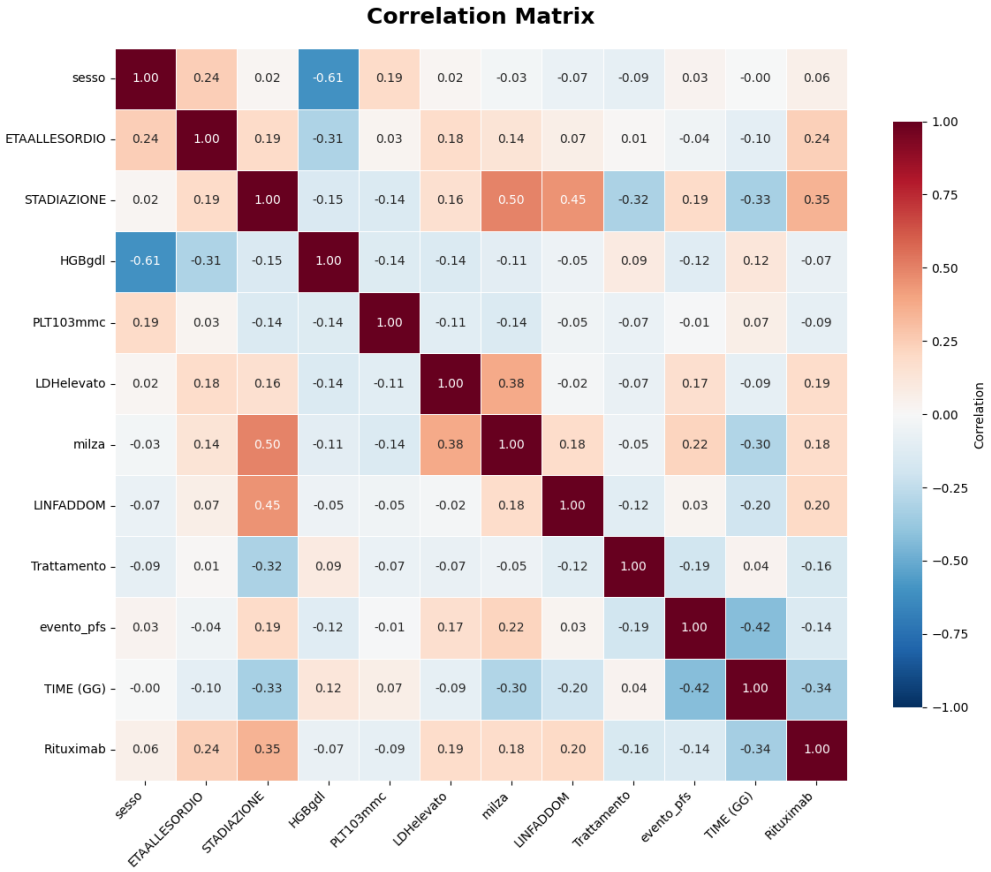


Figure 2.3: Correlation matrix of the original dataset

Chapter 3
Methods and tools

We recall that the main goal of clinical scientists is to understand whether Rituximab should be implemented when treating patients affected by NLPHL. This can be discussed implementing deep learning techniques and analyzing the results. Though, there is a problem when applying these techniques to our dataset: scarcity of data.

The dataset that we are working on contains information about 293 patients. This number is extremely small. The necessary number may depends on many factors, but a minimum of thousands of samples could be necessary to generate a reliable analysis.

To understand the relationship between the use of Rituximab and the effects on patients we are going to need a bigger dataset.

The problem that we are facing is that we do not possess a bigger dataset.

The solution that we present is the use of data augmentation.

The topic that we want to address in this chapter is how to generate synthetic dataset while preserving the privacy of the patients, but at the same time being valuable material for analysis studies. Since we are dealing with sensitive data, this is a fundamental feature to consider.

Privacy concerns is a very discussed issue when working with clinical data. There are not many clinical dataset available, due to this reason. Regarding those that are publicly available, the main example is the MIMIC dataset (Medical Information Mart for Intensive Care), collecting longitudinal data. To make this dataset available, an accurate step of de-identification was made, in order to not let user that uses the dataset be able to retrieve the original sensitive information. Also, only users intending to complete a "Data or Specimens Only Research" are allowed to use the dataset.

We are presenting this example to show how delicate this argument is. Our dataset, however small, contains risky details that cannot be disclosed outside.

To deal with this anonymization problem, the solution can be presented by a synthetic dataset. We need to generate data good enough to be useful, without losing the original statistical information and relationships, and at the same time the user that is looking at synthetic data must not be able to go back to the original patient. So, it's important to balance two features: privacy and resemblance. Please note that one goes against the other. As we increase the resemblance of the generated dataset, the risk of comprising the privacy increases. On the contrary, a dataset where it's impossible to restore even the smallest section of the original version, may not be very accurate in generating values similar to the original ones. A good balance is needed.

The methods that we will discuss in this chapter are the possible solutions to this problem.

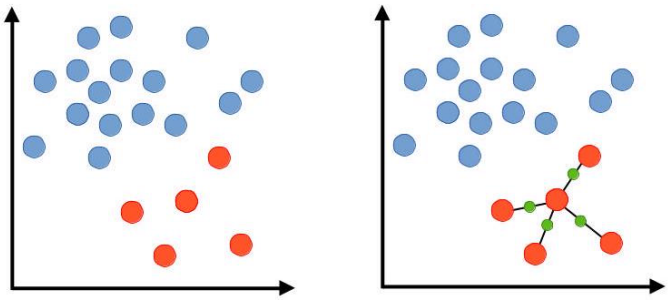


Figure 3.1: This is image shows the approach used by SMOTE algorithm. The blue points corresponds to the majority class, while the orange points are the minority class. The plot on the left is before applying SMOTE, while the plot on the right is after applying SMOTE. The green points in the right plot are the new synthetic points generated by SMOTE. They will be part of the original minority class.

The goal of data augmentation is being able to create new, unseen datasets, starting from the original given dataset, that are different from it, but at the same time the structure and relationships between features are not altered. This means, that the new dataset can be described in the same way as the original one.

In the field of data augmentation, there are different tools that can be implemented. We are going to discuss some of them, to give an overview of the existing methods for data augmentation.

If the input dataset is composed by images, a simple approach is found in techniques like rotating the images, flipping, random cropping, rescaling. This allows us to generate new images.

Regarding tabular data, different methods can be used. First, we will introduce the topic showing two simpler data augmentation models (SMOTE and GMM), while subsequently we will address two more complex synthetic generation methods that create an entire new dataset.

SMOTE and GMM

Synthetic Minority Over-sampling TEchnique (SMOTE) [8] is a technique developed to solve the problem of minority classes. A minority class is a portion of the dataset that is less represented. For example, it can be the case of a study collecting data about patients with a specific disease, where the data were collected during school time, so not many younger people could participate, but the study was intended to be conducted among all ages. In this case, it's probable that the applied model would learn how to treat the disease based on an higher average age. To create a good balance, it would be necessary to increase the number of samples in the minority class.

SMOTE increases the size of the class by generating new synthetic samples. This method takes the minority class and introduce the new samples along the linesegments between the existing samples. The number of nearest neighbors can be chosen [8]. An example is shown in Figure 3.1.

Gaussian Mixture Models are a method that learns the distribution of the original dataset and tries to recreate another one with the same distribution. The model assumes that the given data points are the result of a mixture of a finite number of Gaussian densities. [3] The model find the best parameter set and to fit the joint probability of the density.

Then, it can generate synthetic data similar to that density. [24]

3.1 Generative Models for synthetic data

In this section we are going to present the major available methods for generating synthetic datasets, describing in details the two methods that we implemented. Additionally, two Python tools via which the methods can be used can be found in the appendix.

There are specific tools that can be trained to generate synthetic dataset.

These models are used in many different environments, not only in health-care applications, but also in energy consumption, finance, public sector, smart-home activities and many others. Basically, synthetic dataset can be implemented to solve a huge number of different problems in different fields. We will be interested in health-care applications, where data are usually represented in a tabular way.

We already explored the need for use of synthetic data in our research problem, so now we will show what are the main approaches that can be implemented to do so.

Following the approach presented in paper [17], there are different types of methods to create synthetic generated datasets. The two main types are: classical approaches and deep learning models.

The classical approaches include baseline methods, statistical methods and machine learning models.

Baseline methods refer to methods that just use an anonymisation technique, fixing values in the original dataset. The model generates synthetic data by memorizing the relationships of the real features [17]. The statistical models capture correlation between features in the original dataset and try to simulate them. Machine learning models refer to unsupervised models that learn how to recreate synthetic data. This is done time-sequentially, so the model predicts the next-step group of patients that will be considered the synthetic dataset.

The deep learning methods refers to complex neural networks, and they include: Autoencoders (AEs), Generative Adversarial Networks (GANs) and Ensembles.

AEs are composed by two steps: the first one is an encoder, then there is a decoder. The encoder compresses the data, while the decoder reconstructs the data starting from the compressed representation. This is very efficient especially for categorical and binary features.

GANs became rapidly very popular. Their main application is on image generation, where their robustness is best shown, but the model can be applied also for the generation of tabular data. GANs are composed by two subsequent neural networks. The first one is a generator, that tries to generate synthetic data, while the second one is a discriminator whose job is to decide if the data produced by the generator is real or synthetic. In the case the discriminator has a fifty percent probability of success (i.e., equivalent to random guessing), the balance between the two is good.

Ensembles are a technique that combines multiple deep learning methods to generate a unique result. Though, the results produced by ensembles are not equally good as for other deep learning tecnhiques.

Figure 3.2 shows the different existing methods explored by the paper [17].

Not included in their study [17], Diffusion Models are a valid alternative to the previously said methods. They are the most used in synthetic data generation, because of their stability and strong model capacity [19], and they have produced very promising results in the last years gaining lot of attention. Diffusion Models work in two steps: first, they gradually add Gaussian noise [31] to the input data until obtaining pure Gaussian noise,

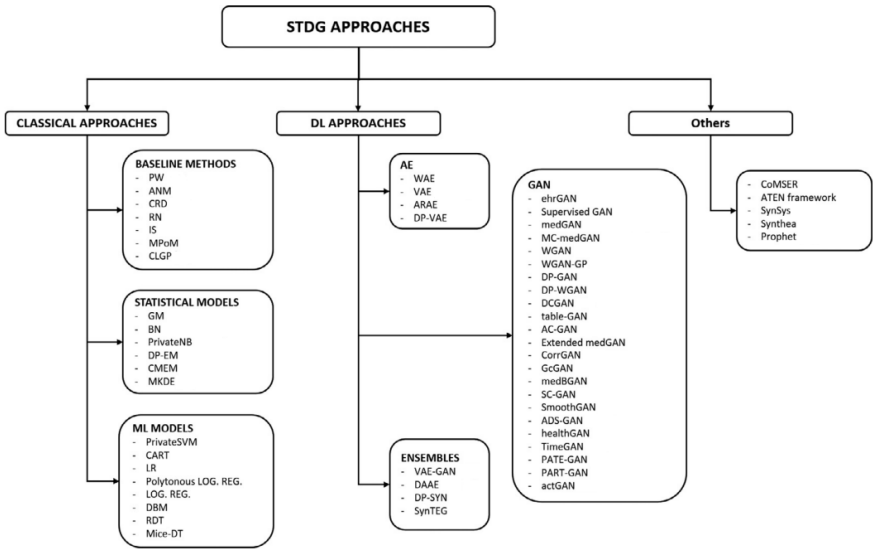


Figure 3.2: Existing tools for generation of synthetic dataset. This image was taken from paper [17]. For more information, refer to the original paper.

then they learns to recreate the original data from that noise.

The first interest of this project was in analyzing in details the performance of a GAN given their popularity within generative techniques and their applications in many different fields. Given the reduced dimension of the dataset of interest, the second interest was in experimenting Diffusion Models, to see if their performance can overcome the performance of GANs.

Synthetic Generative Procedure

In this project we used two different approaches for the synthetic generation of our Lymphocyte Predominant Hodgkin Lymphoma dataset provided by the San Matteo Hospital of Pavia. The first approach is a HealthGAN devloped by the author in paper [34], while the second one is a Tabular Denoising Diffusion Probabilistic Model (TabDDPM), developed by the author in paper [22].

In both cases, we start with the preprocessed dataset that we described in Chapter 2. In the HealthGAN approach, the dataset is converted to SDV format, then fed into the network and a synthetic dataset is produced. A final re-conversion from SDV format to original shape was necessary.

For the TabDDPM, the dataset undergoes a logarithmic function, and then the numerical variables are standardized, while the categorical ones are kept unchanged. The Diffusion Model generates synthetic dataset, and then it is converted back to the original shape through a de-standardization for the numerical columns and an exponential function is applied to all the columns.

After the generation of the synthetic dataset, the right censoring that we described in Chapter 2 is applied.

3.1.1 GAN

A Generative Adversarial Network is a deep learning method for generation composed by two neural networks used for unsupervised learning.

Within the generative models, we can distinguish between models that explicitly retrieve the original density of the data, and model that do it implicitly [10]. GANs fall into the second category: this means that no distribution density is defined and the models use a stochastic procedure that directly generates the new data. This can usually produce more realistic results.

First GAN implementation was proposed in 2014 by Ian Goodfellow [13]. After this first approach, many others were developed and tested [27]. Now we are going to describe the general procedure of a GAN based synthetic data generation.

The two neural networks composing a GAN are: the Generator, which generates new data, and the Discriminator, whose work is to understand if the data presented from the Generator is real or synthetic, given the real dataset as reference. The Generator never encounters the original dataset, but it learns its features and distributions by interacting with the Discriminator.

They are playing an adversarial game, where the Generator learns to map random noise vectors to some synthetic samples that mimic the original distributions. Then, the Discriminator's work is to distinguish those new samples from the original ones. As this game goes on, the Generator progressively improves its output until the new data becomes indistinguishable from the original.

The process is represented in Figure 3.3.

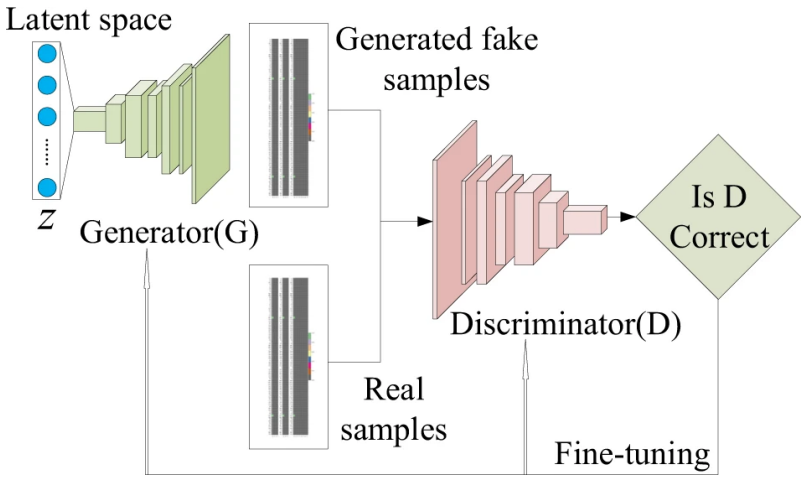


Figure 3.3: Architecture of a GAN. This image was taken from [18].

As an example, Figure 3.4 shows the simplified process of a GAN for image generation. At first, two processes work in parallel. The original dataset is decoded and the information is extracted. The generator starts looking at random noise (no information about the real dataset) and attempts to recreate the original data. The first attempts are basically random guessing, since the Generator does not know anything about how the data should look like. The Discriminator receives the results of these trials together with the original dataset. Its job is to decide if the data created by the Generator is from the real dataset or generated synthetically. In the first steps, the Discriminator will always see data that look nothing like the original dataset, so it will mark them as synthetic. As this adversarial game continues, the Generator gets better at generating data similar to the original dataset. The ideal situation is reaching a point where the Discriminator is no longer able to say if the presented data is real or synthetic.

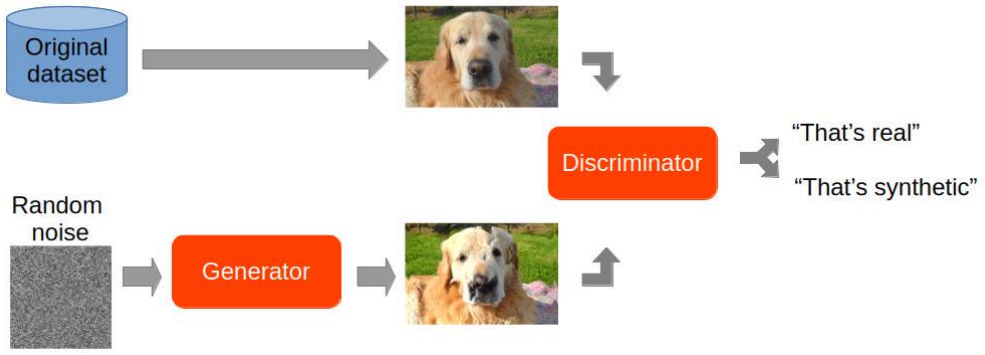


Figure 3.4: GAN process

Despite the strengths of GANs, some issues have been found in literature.

The first problem that GANs suffer from refers to the convergence instability, while another clear problem is mode collapse [37].

The first one happens when encountering a vanishing gradient of the generator whenever the best discriminator is found. The second one happens when the model is not able to capture the diversity of the original dataset and produces samples similar between them. The classic example of mode collapse is a binary column of the original dataset that is converted to a single-value column in the synthetic generated dataset.

For a rapid implementation of a GAN, the Pygan library in Python can be used. The use of the library is described in the appendix.

Implementation

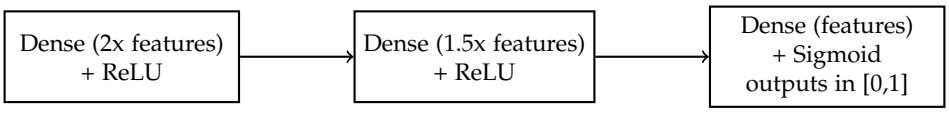
We implemented a slightly different version of the standard GAN, i.e. Wasserstein GAN-based method called HealthGAN [34]. The difference from the standard GAN can be found in the implementation of the Wasserstein loss to help stabilize the training part. A second difference exists in the introduction of a gradient penalty. Eventually, the HealthGAN may still generate poor samples because the training can be stuck due to vanishing or exploding gradients. A gradient penalty penalizes gradients with large norm values. The standard GAN would use weight clipping to solve this issue.

Additionally, the Discriminator in our implementation does not produce a binary output, traditionally obtained with a Sigmoid activation function, but it outputs a value that can be either positive or negative. If it's positive, it means that the Discriminator thinks the data is real; if it's negative, the Discriminator thinks it's synthetic. The magnitude of this value refers to the confidence that the Discriminator has when producing the result.

Our implementation of the HealthGAN is taken from [34].

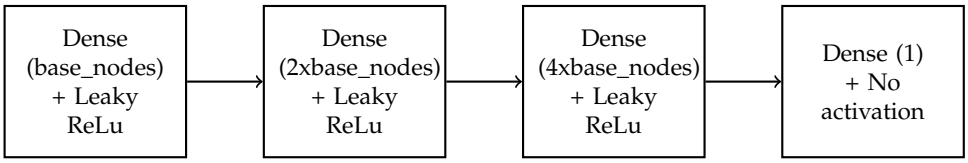
Before starting with the actual GAN, a preprocessing step is necessary. The original data is encoded and transformed into a continuous distribution between 0 and 1. The categorical values are mapped into this range using truncated normal distribution and the numerical ones are normalized within the same range. This creates an SDV format necessary for feeding the data into the network.

Then, we have the real HealthGAN with gradient penalty process. The SDV dataset is split into train and test, using a 90-10 division. A batch generator is created, in order to yield properly sized batches. At this point, the generator starts generating data. Its structure is composed by three layers:



The output of the Generator is a synthetic dataset ready to be analyzed by the Discriminator.

The Discriminator is a neural network composed by four layers:



The output of the Discriminator is a raw score regarding the dataset presented by the Generator.

This score is kept under control by the gradient, measuring how much the response of the Discriminator would change if the presented data were slightly modified. If the gradient is high, this means that, for a small change in the data, there is a big change in the Discriminator decision. On the contrary, a flat gradient suggests that the Discriminator does not vary much when the presented data is modified. The gradient penalty penalizes slopes that are far from 1.

At the end, we have a decoding phase. The values are mapped back to their original range.

3.1.2 Diffusion Model

Denoising Diffusion Probabilistic Models (DDPM) [35] are now object of study for many researches in the generative models field, since they can often produce better results than the alternative approaches. Initially, their power was shown in image generation and Natural Language Processing.

DDPM are generative models that work through a Markov process (forward and reverse) [22]. The forward steps gradually adds noise to the original dataset, until reaching a pure noise state. The reverse process works the opposite way and tries to recover the original data given the pure noise state. An example is shown in Figure 3.6, where the input data is an image. Markow process means that each step depends exclusively from the previous step. This can be described with the probability formula $P(x_t|x_{t-1}, x_{t-2}, ..., x_0) = P(x_t|x_{t-1})$.

The structure of the Diffusion Model can be seen in Figure 3.5.

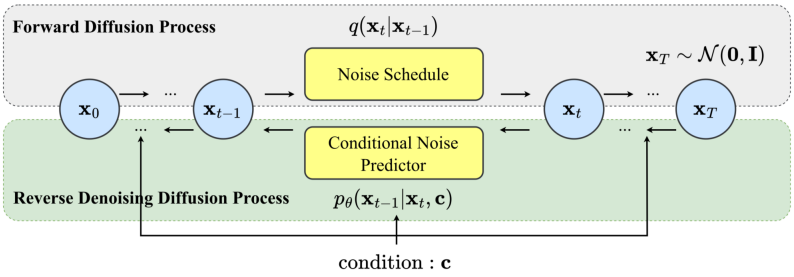


Figure 3.5: Diffusion Model architecture. This image was taken from [15]

This process is made via a Gaussian diffusion for numerical features and a Multinomial diffusion for categorical features. Gaussian diffusion corrupts the values adding a Gaussian noise to the data, while Multinomial diffusion works gradually mixing the categorical value until the original data is not visible.

A study introduced in 2024 [22] released a new version of DDPM, addressing the problem of generating tabular synthetic dataset. This model is called Tabular Denoising Diffusion Probabilistic Model (TabDDPM). This method can be applied to mixed data types, either numerical or categorical. The paper [22] shows the great performances of this model, outperforming the other existing approaches, inluding GAN-based models.

A simple implementation of a Diffusion Model can be used via the Pytorch library in Python. The use of the library is described in the appendix.

Implementation

Our implementation of the TabDDPM is taken from [22].

The original code provided a data preprocessing process using the gaussian quantile transformation. In our project, for better control of the process, we decided to use the same preprocessing that we had already use for the HealthGAN, with some modification, and silenciate the preprocessing provided in the TabDDPM code.

Our modified preprocessing starts with applying the logarithmic function to the data. This ensures a more manageable distribution when converted. After the logarithmic function, the mean and standard deviation of the data are stored. Then, numerical features are normalized with z-score in order to have mean equal to 0 and standard deviation equal to 1. Note that with this transformation the data are not converted to proper SDV format. In the case of GAN, SDV format was necessary for running the code, while in our diffusion model code this specification was not necessary, since the model is able to read data in different ranges.

After this, the data are fed into the network. Through a training loop of noising and denoising, the Generator learns to improve, until a good balance with the Discriminator is reached. Then, in the sampling phase, the new synthetic dataset is generated.

Now, the produced numerical samples are normalized with mean equal to 0 and standard deviation equal to 1. This is what we expected, since we inputed data in that shape. To go back to the original distribution range, we first converted them using as reference the mean and standard deviation that we had stored at the beginning after the logarithmic function. Ultimately, we applied the exponential function to recompose the original data shape.

This was just for the numerical columns. The categorical columns did not use any preprocessing.

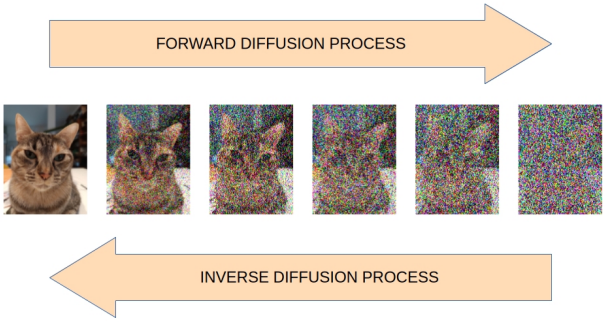


Figure 3.6: Example of a diffusion process

Diagram 3.7 represents the steps of the implemented model.

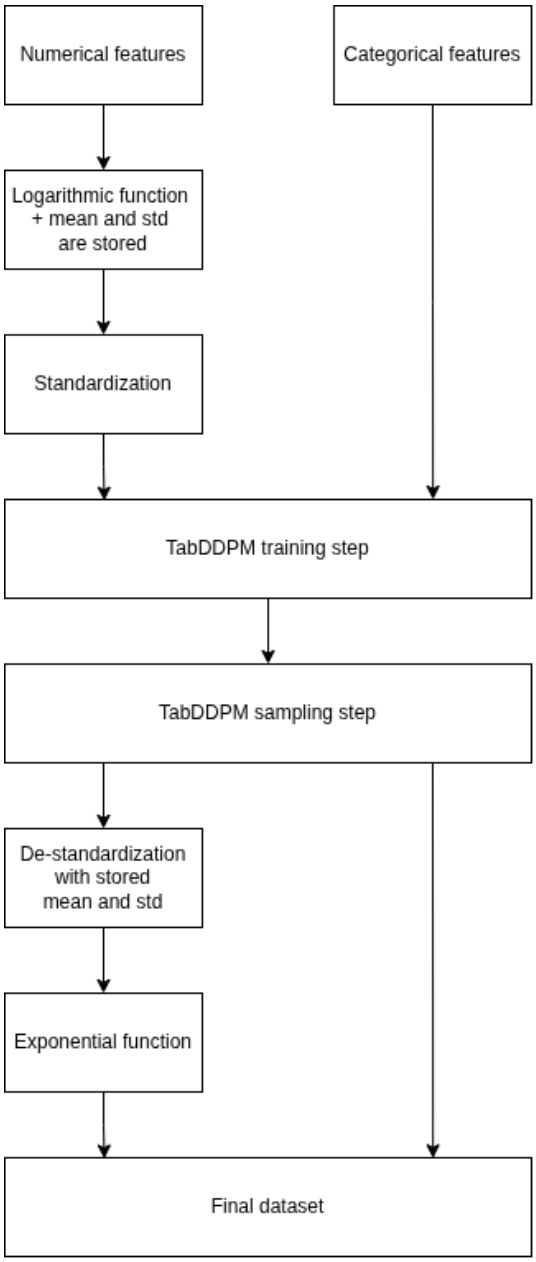


Figure 3.7: Our TabDDPM process

3.2 Evaluation methods for synthetic data

After the generation of synthetic dataset with the methods described in the previous section, we need to perform an accurate exploration of the generated dataset in order to establish their reliability and the risks that it can generate (in terms of privacy and false reliability). Based on existing literature, there are three aspects that have to be analyzed. They are: resemblance, privacy, utility.

The resemblance is a metric that acts as a preliminary quality control. It answers the question of whether the generated synthetic data have preserved the original structure. This result alone is not enough, but a bad score in this metric suggests that the synthetic dataset cannot be trusted.

The privacy metrics refers to the possibility of being able to retrieve the original dataset, given the synthetic one. This answer the question of whether the original data are exposed. This is particularly important in the medical field, where sensitive data must not suffer the risk of being exposed. Another way is mimicking cyberattacks: this consists in analyzing how well an attacker could reconstruct the original dataset from the synthetic data, given a small portion of the real dataset.

The utility feature can be explored by performing Train on Real Test on Real (TRTR) and Train on Synthetic Test on Real (TSTR) methods [29]. Some machine learning models are trained on a subset of the real dataset or a substet of the synthetic data, respectively, and tested on a subset of the real dataset (in the case of TRTR, different from the one used for training). Consequently, a classification metric can be used to assess the two performances.

There are already some existing tools that explore these three metrics [32], but in order to have an even more efficient tool, the following aspects have to appear [32]:

- it must analyze resemblance, privacy and utility;
- it must have a user-friendly graphical interface and be easy to use. In this way both programmers and clinicians can use the same tool;
- it must provide a report with the results;
- it must allow for benchmarking.

Given this, a new tool was developed: SynthRO [32], Synthetic data Rank and Order. This is the tool that we are going to use to analyze the quality of our synthetic dataset.

Additionally, considering the intentions of the clinical staff with the synthetic dataset, survival curves [11] will be generated. Survival curves are used to evaluate the utility of the dataset.

We will start discussing SynhtRO, then we will move to describe survival analysis.

3.3 SynthRO

SynthRO [32] (Synthetic data Rank and Order) is a tool developed in Department of Electrical, Computer and Biomedical Engineering, University of Pavia, Italy.

The creation of this tool had four main objectives: perform an analysis based on resemblance, privacy and utility metrics, have a user-friendly graphic interface, provide reports for each analysis and allow for benchmarking.

SynthRO is made with Python language. In particular, the Dash package was used, perfect for web data visualization [1]. The code of SynthRO can be found at: [2]

When SynthRO is launched, we can see five sections: Load Data, Resemblance, Utility, Privacy and Benchmarking. The structure of the tool is illustrated in Image 3.8.

[1] `https://github.com/plotly/dash`
[2] `https://github.com/bmi-labmedinfo/SynthRO/tree/main`

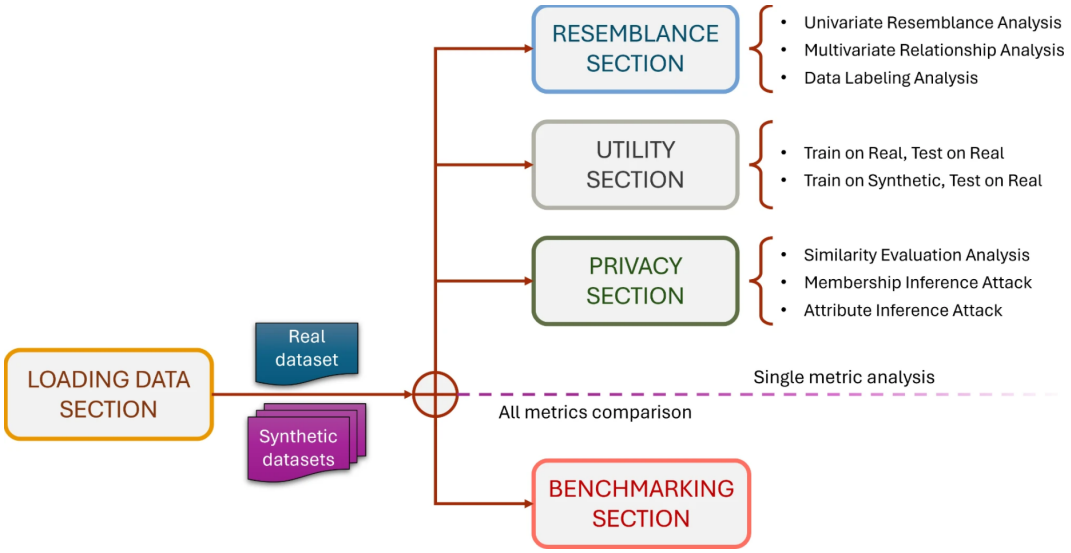


Figure 3.8: Structure of the SynthRO tool. This image was taken from the original paper [32]

Load data

The first step is uploading the two datasets (the real one and the synthetic generated one) in the first section. Right after this, it is necessary to define the type of the columns contained in the dataset (numerical or categorical). To do that, a text file with that information can be uploaded, or the type of the columns can be chosen via a drop-down menu. In our case, we did not upload a file but used the menu.

The next sections refer to the three main analysis metrics: resemblance, utility, privacy.

Resemblance

The resemblance metric can here be performed with three methods: univariate resemblance analysis (URA), multivariate resemblance analysis (MRA) and data labelling analysis (DLA).

The univariate resemblance analysis shows if the univariate statistical properties of the original dataset are conserved also in the synthetic dataset, analyzing one feature at a time. In this section you can perform statistical tests and calculate distance metrics.

For the numerical columns, you are able to choose if you want to perform a Kolmogorov-Smirnov test (used to compare the cumulative distributions), Student t-test (comparing the mean of the two groups) or Mann Whitney u-test (comparing the median). For the categorical columns, a chi-squared test is available, to check association between the columns. For the distance calculations, you can choose between cosine distance, Jensen-Shannon distance and Wesserstein distance.

The multivariate resemblance analysis, instead, analyzes this similarity taking into consideration multiple features at the same time, creating a multidimensional context. Again, the goal is to answer whether the real statistical properties are replicated in the synthetic dataset. In this analysis you can display correlation matrices. For numerical columns, Pairwise Pearson Correlation is proposed, while normalized contingency tables are available for the categorical columns. You can choose to either display two matrices (real and synthetic) or one single matrix containing the difference between the two.

In this section you can also compare the presence of outliers via a graph and compare Principal Component Analysis.

Finally, you can also create a UMAP visualization, capturing the global variability in a two-dimensional world. For a good synthetic dataset, we expect the two plots to be similar.

The last operation that you can do in the resemblance section is data labelling analysis. Here, some classifiers are trained to recognize if the data are real or synthetic. You can select one or more classifiers to perform the test. After running it, you will see a summary of evaluation metrics of each selected classifier, together with a representing plot.

Utility

Next metric is the utility metric. In this part you can compare performances (derived from different classifiers) on real dataset versus on synthetic dataset. You select a target class (one of the features) and choose between one of the said models. Then, you will see the performance of Train on Real Test on Real, compared to the performance of Train on Synthetic Test on Real, via a correlation matrix.

Privacy

The last metric is the privacy metric. Here, we can check if the original dataset has a risk of being exposed, given the synthetic dataset.

The first method here is to to a similarity evaluation analysis, that is computing the distance between real and synthetic data. .

The second methods simulates cyberattacks, given a subset of the original dataset.

In every part of these evaluations, you are always able to download a pdf report with relative graphs and tables, via the button "Download Report".

3.3.1 Benchmarking

The last part is related to benchmarking. This was introduced to be able to share the results in anticipation of future works. Here, you can combine the three metrics analyzed before, assigning different weights to each part. Each metric will have a score between 0 and 1. Then, the scores are multiplied by their own weights, and then summed together. A lower score will represent a better result.

3.4 Survival Analysis

Survival analysis is a way of representing the time passing between an initial event and a final event, which can either happen or not [11]. In the clinical domain, this may correspond to the time between the diagnosis of the disease and the death of the patient. Even if death is usually the considered event, this one can be identified by any other binary event. Survival curves are used to describe the survival of a single group of patients, but they can also be used to compare multiple groups, for example groups of patients receiving different types of treatments. In that case, the goal may be to understand which treatment is the most effective.

For the goal of survival analysis, we need to address the censoring concept. A patient may show "loss to follow-up", i.e. the information about the patient is interrupted before the end of the study, then before suffering the event. Anyway, those patients influence the whole outcome of the survival analysis. To conduct the analysis, we need to assume that

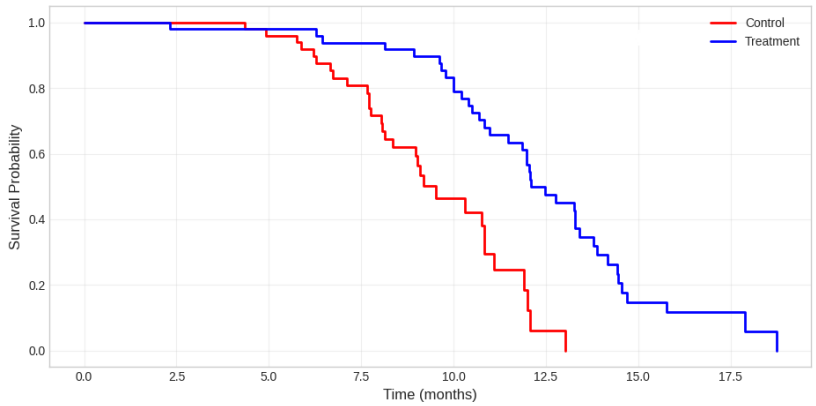


Figure 3.9: Example of a survival analysis via Kaplan Meier method. The reference dataset is composed by patients that undergo a treatment (treatment group) and patients that don't (control group). This plot represents, at each step in time (in months), the percentage of patients that haven't experienced the event of interest. In this case, the entire control group experiences the event of interest in the first 13 months, while the treatment group experiences it in the first 19 months.

the censoring is non-informative: the censoring of the patient is not related to the probability of the event occurring.

One of the main approaches to create a survival analysis is the Kaplan and Meier method [5]. This technique considers the patients that have not experienced the event yet at every time in point and represent them in a plot, where the x-axis is the time and the y-axis is the proportion of patients that haven't experienced it.

Figure 3.9 represents a possible Kaplan and Meier survival analysis.

However, survival curves are not sufficient by themselves. Even if two survival curves are slightly different between themselves, it's not clear if that's because of a meaningful difference in the patients or if the difference arises by chance. To exactly understand the cause, the log-rank test can be applied [6]. Visually comparing the two curves suggests that you compare specific time steps, while the logrank test takes into account the entire curves. The logrank test is used for testing the null hypothesis of no difference between the two cohorts. For both groups, the probabilities of experiencing the event is calculated for each time based on the number of experienced event until that specific time. With a test statistic with χ^2 distribution you can test the null hypothesis and calculate the p-value [6]. If the p-value is smaller than 0.05, this implies that there is a significant difference between the two curves [11].

Chapter 4
Results

In this chapter, the results obtained will be presented. We are going to describe the datasets obtained with the synthetic generation methods: first with GAN and later with TabDDPM. Each of these results will be analyzed through two evaluation methods: SynthRO and survival analysis. We will approach the resemblance metric through SynthRO and the utility metric via survival analysis.

Before reading this chapter, it's important that the reader keeps in mind that the entire evaluation process is made comparing the resulting synthetic dataset with the same initial dataset that was fed into the generating model. So, the datasets are evaluated on the same dataset they were trained on.

We remind that this is not the optimal approach. The choice of proceeding in this way was caused by the small size of the dataset. Considering that the original dataset contained information about just 293 patients, even splitting that dataset in training and test datasets was considered risky.

More complete studies [2] have shown that the results obtained with a single train and test split can be unreliable when the dataset size is not sufficient. Splitting the dataset reduces its size and that can compromise the reliability of the results obtained from a model trained with that dataset. Due to this reason, even a cross-validation would lead to untrustworthy results. The cited study [2] was made testing a dataset of 1000 samples.

We proceeded using the same dataset for training the models and evaluating the results keeping in mind that the results obtained may be a little too optimistic. Either ways, we obtained very different results from generating synthetic data with GAN or TabDDPM, so we didn't suffer from this side effect in our analysis.

For each one of GAN and TabDDPM, a single synthetic generated dataset was considered and analyzed. Then, that same dataset is split into two subsets: one containing patients treated with Rituximab, and one for patients treated without Rituximab. These two subsets are tested against the corresponding subset of the original, right censored dataset.

This division was made in order to better understand the correlation between the two cohorts and analyze the difference between them.

Before starting with the analysis of the synthetic datasets, we studied the original dataset and the two corresponding subsets (Rituximab and no-Rituximab) to check if substantial differences are present. This study was made via SynthRO for the resemblance metrics, and survival analysis for the utility metrics.

In this chapter there will be four sections: the first one related to the analysis of the original dataset, the second and the third ones related to the GAN ad TabDDPM synthetic

generated dataset respectively and the last one will compare the two synthetic datasets.

Experiments were run on a computer with an Intel Core i7 processor, with integrated graphics Intel Iris Xe, running Ubuntu 25.10.

4.1 Analysis of the original dataset

For the reason explained in Chapter 2.2, we applied right censoring at 1825 days to the original dataset.

With a simple R code, we implemented a Table One [28] to retrieve the statistical information of the dataset.

On SynthRO, we uploaded the entire original dataset as "real dataset" and the subsets composed, respectively, of Rituximab and no-Rituximab patients as "Synthetic datasets". We proceed with an analysis of the Resemblance metric.

A summary of the retrieved information is shown in Table 2.2.

Then, we generated correlation matrices to observe the relationship between the features, obtained with a Pairwise Pearson correlation matrix for the numerical features and a normalized contingency table for the categorical features. They are displayed in Figure 4.1. These three matrices will be used as reference for the analysis of the GAN and TabDDPM results, where the difference between the synthetic ones and the original one will be showed.

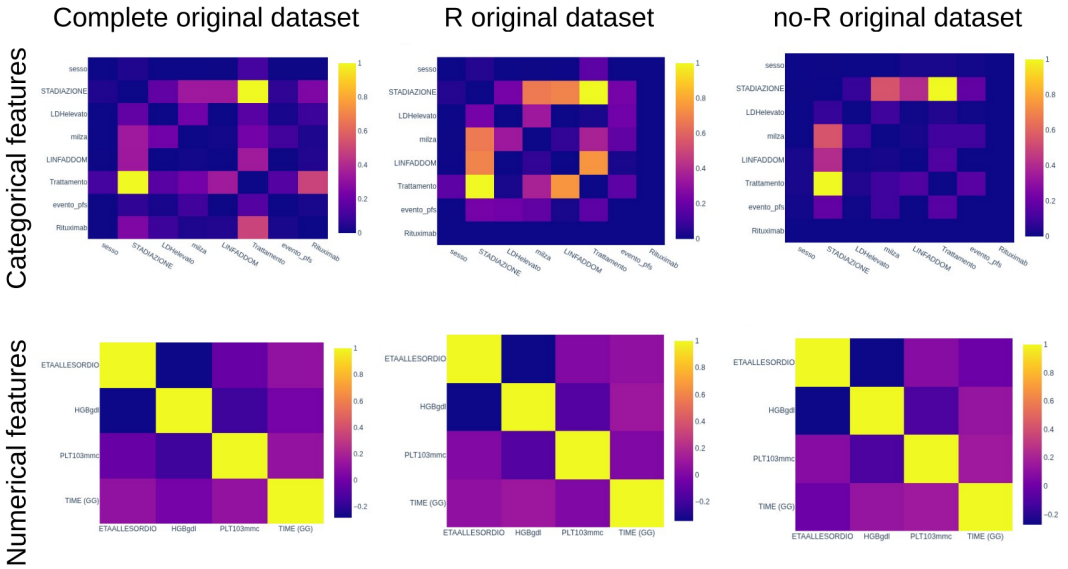


Figure 4.1: Correlation matrices of the original dataset

For an analysis of the utility metric, we created survival curves, as described in Chapter 3. The results are shown in Figure 4.2. Three lines are plotted, corresponding to the three analyzed datasets. We can see that there is a difference in the behavior of Rituximab patients compared to the no-Rituximab patients, resulting in the Rituximab patients being more likely to experience the disease progression later in time, respect to the no-Rituximab patients.

We are going to keep in mind these differences, as we continue with the analysis of the synthetic datasets.

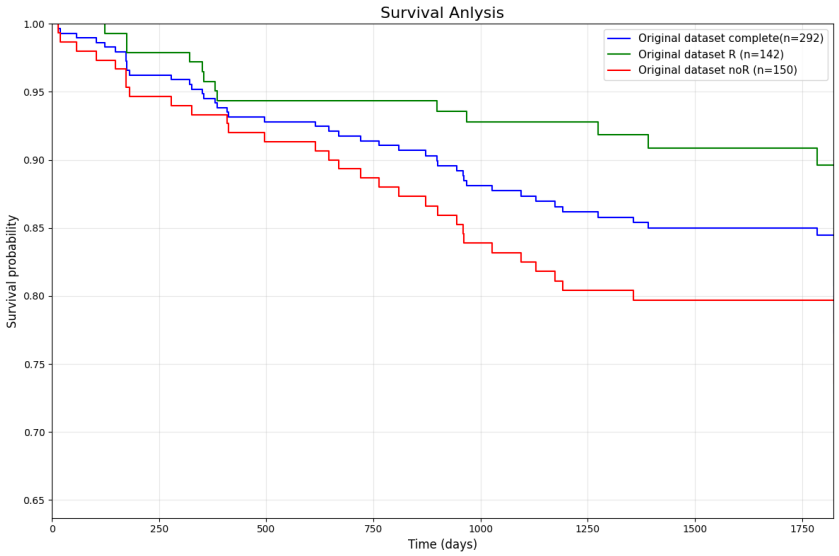


Figure 4.2: Survival curves for the original dataset. The blue line corresponds to the behavior of the entire dataset. The green and the red lines represent respectively the subsets composed just by the Rituximab patients, and the no-Rituximab patients.

4.2 Analysis of GAN results

Running 10000 epochs and generating 10 datasets composed by 1000 samples each, the model takes 1 minutes and 38 seconds.

As a result, we considered one of the 10 generated datasets after the GAN generation. The dataset is composed of 1000 samples. For the reason explained in Chapter 2.2, we applied right censoring at 1825 days. This resulted in the majority of the data being censored. The data is then split into a dataset containing only Rituximab patients and another one with no-Rituximab patients.

Even before looking into the details, it could be seen that the model suffered from mode collapse. It was clear in the binary columns, where the proportions were stretched to the point that some columns contained almost just one value. The model was not able to capture the full diversity of the original data and generate enough diversity in the synthetic data. This was probably due to the smallness of the dataset and its complexity.

As for the original dataset, we proceed with retrieving some statistical info and the p-values. This is shown in Table 4.1

We can see the problems of mode collapse within the categorical columns, confronting this table with the Table 2.2. To show an example in the Rituximab subgroup, the *LINFADDOM* column dropped from ('1': 55.63%, '0': 44.37%) to ('0': 95.40%, '1': 4.60%). This also happened for *MILZA*, *LDHelevato*, *sesso*.

To compare with the original dataset, we developed correlation matrices indicating the difference in absolute value between two correlation matrices: real and synthetic. This is shown in Figure 4.4.

Additionally, the UMAP analysis is shown in Figure 4.6. It is clear to see that the synthetic dataset resembles the original one without capturing all its diversity but just a minor part.

		Overall	No-Rituximab	Rituximab	p-value
n		1000	369	631	
sesso (%)	0	989 (98.9)	365 (98.9)	624 (98.9)	1.000
	1	11 (1.1)	4 (1.1)	7 (1.1)	
ETAALLESORDIO (mean (SD))		59.22 (7.26)	59.68 (7.42)	58.95 (7.16)	0.125
STADIAZIONE (%)	1	621 (62.1)	169 (45.8)	452 (71.6)	<0.001
	2	337 (33.7)	197 (53.4)	140 (22.2)	
	3	42 (4.2)	3 (0.8)	39 (6.2)	
HGBgdl (mean (SD))		15.58 (0.90)	15.61 (0.86)	15.56 (0.93)	0.402
PLT103mmc (mean (SD))		204.18 (51.66)	190.33 (49.62)	212.27 (51.14)	<0.001
LDHelevato (%)	0	986 (98.6)	365 (98.9)	621 (98.4)	0.710
	1	14 (1.4)	4 (1.1)	10 (1.6)	
milza (%)	0	994 (99.4)	369 (100.0)	625 (99.0)	0.146
	1	6 (0.6)	0 (0.0)	6 (1.0)	
LINFADDOM (%)	0	958 (95.8)	356 (96.5)	602 (95.4)	0.514
	1	42 (4.2)	13 (3.5)	29 (4.6)	
Trattamento (%)	1	544 (54.4)	232 (62.9)	312 (49.4)	<0.001
	2	4 (0.4)	1 (0.3)	3 (0.5)	
	3	84 (8.4)	17 (4.6)	67 (10.6)	
	5	368 (36.8)	119 (32.2)	249 (39.5)	
evento_pfs (%)	0	896 (89.6)	337 (91.3)	559 (88.6)	0.207
	1	104 (10.4)	32 (8.7)	72 (11.4)	
TIME..GG. (mean (SD))		1820.21 (43.71)	1819.63 (44.22)	1820.55 (43.43)	0.749

Table 4.1: Statistical analysis of the GAN synthetic generated dataset

When we got the point of doing the survival analysis, we identified another major problem: there were no events at all before 1825 days in the synthetic generated dataset. All the *evento_pfs* with value 1 (affirmative) were of those patients with loss of follow-up, so they do not participate in the survival curve. Because of this, survival curves were not useful, since the plotted line would be perfectly horizontal until 1825 days, both for the Rituximab and no-Rituximab patients.

4.3 Analysis of TabDDPM results

The TabDDPM procedure generated one single dataset composed of 1000 synthetic patients, taking 3 minutes and 10 seconds. After the generation, an initial screening is done: extreme values out of the initial range are deleted, for the columns *HGBgdl*, *PLT103mmc* and *ETAALLESORDIO*. The final number of patients is 920. Right censoring is applied and the main dataset is split in two subsets: Rituximab and no-Rituximab. We proceeded before.

For the resemblance metric, we prepared a table with statistical information (Table 4.2) and the difference in correlation matrices as we did for the GAN dataset (Figure 4.5).

		Overall	No-Rituximab	Rituximab	p-value
n		920	461	459	
sesso (%)	0	734 (79.8)	383 (83.1)	351 (76.5)	0.016
	1	186 (20.2)	78 (16.9)	108 (23.5)	
ETAALLESORDIO (mean (SD))		44.04 (14.11)	47.61 (14.39)	40.46 (12.88)	<0.001
STADIAZIONE (%)	1	261 (28.4)	111 (24.1)	150 (32.7)	<0.001
	2	170 (18.5)	91 (19.7)	79 (17.2)	
	3	122 (13.3)	47 (10.2)	75 (16.3)	
	4	367 (39.9)	212 (46.0)	155 (33.8)	
HGBgdl (mean (SD))		14.13 (1.49)	14.20 (1.48)	14.06 (1.50)	0.137
PLT103mmc (mean (SD))		244.03 (81.86)	249.76 (82.96)	238.27 (80.41)	0.033
LDHelevato (%)	0	627 (68.2)	300 (65.1)	327 (71.2)	0.053
	1	293 (31.8)	161 (34.9)	132 (28.8)	
milza (%)	0	803 (87.3)	412 (89.4)	391 (85.2)	0.071
	1	117 (12.7)	49 (10.6)	68 (14.8)	
LINFADDOM (%)	0	414 (45.0)	170 (36.9)	244 (53.2)	<0.001
	1	506 (55.0)	291 (63.1)	215 (46.8)	
Trattamento (%)	1	74 (8.0)	31 (6.7)	43 (9.4)	<0.001
	2	255 (27.7)	119 (25.8)	136 (29.6)	
	3	470 (51.1)	286 (62.0)	184 (40.1)	
	5	121 (13.2)	25 (5.4)	96 (20.9)	
evento_pfs (%)	0	782 (85.0)	418 (90.7)	364 (79.3)	<0.001
	1	138 (15.0)	43 (9.3)	95 (20.7)	
TIME..GG. (mean (SD))		1557.60 (481.50)	1616.86 (440.49)	1498.08 (513.08)	<0.001

Table 4.2: Statistical analysis of the TabDDPM synthetic generated dataset

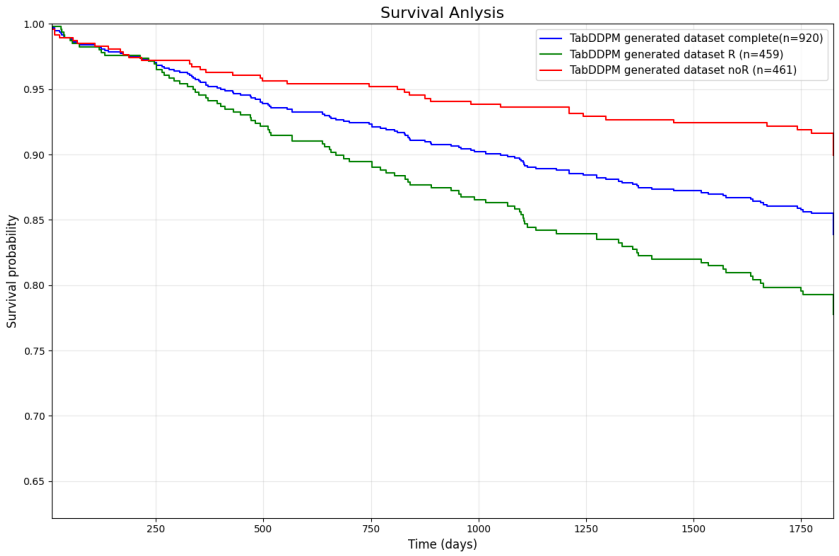


Figure 4.3: Survival curves for TabDDPM synthetic dataset.

Comparing to the original dataset, we can see how the distributions of the features are maintained. The model was able to generate data without falling into mode collapse.

Additionally, the UMAP analysis is shown in Figure 4.7, showing good performance in resembling the original dataset.

For the utility metric, survival curves were studied. They can be seen in Figure 4.3. Three lines are plotted, corresponding to the three analyzed datasets. We can see that there is a difference in the behavior of Rituximab patients compared to the no-Rituximab patients, resulting in the no-Rituximab patients being more likely to experience the disease progression later in time, respect to the Rituximab patients. This is exactly the opposite than in the original dataset, and this is going to be the main problem of the TabDDPM generated dataset. Since the main goal of the project is to analyze the effectiveness of Rituximab on the patients, it is fundamental that a synthetic dataset respects the behavior of the original dataset, under this point of view.

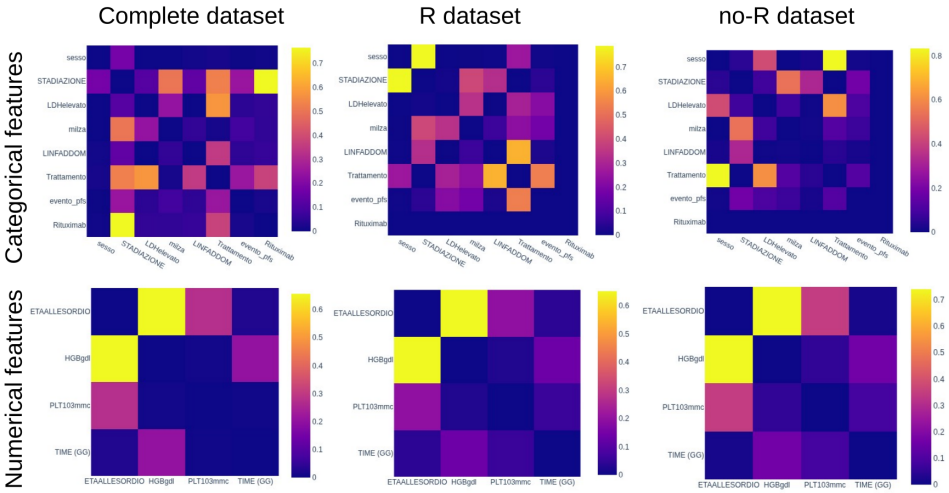


Figure 4.4: Difference in correlation matrix between original dataset and GAN generated dataset

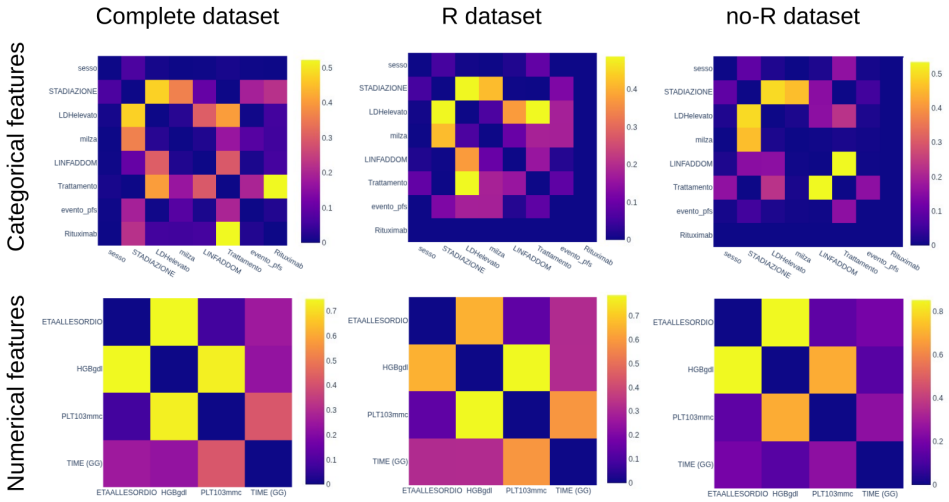


Figure 4.5: Difference in correlation matrix between original dataset and TabDDPM generated dataset

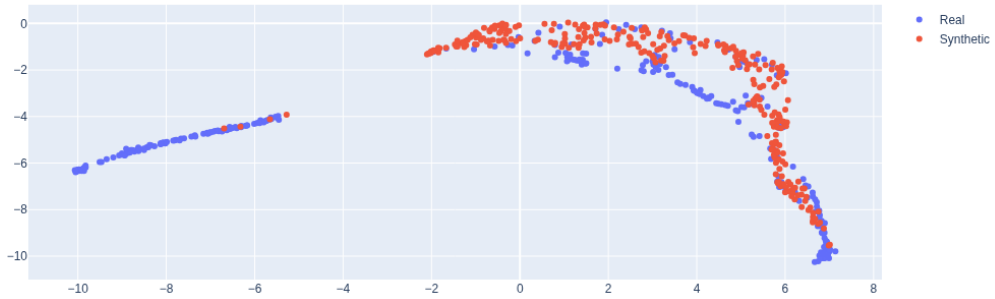


Figure 4.6: UMAP analysis of the real dataset and the GAN generated dataset

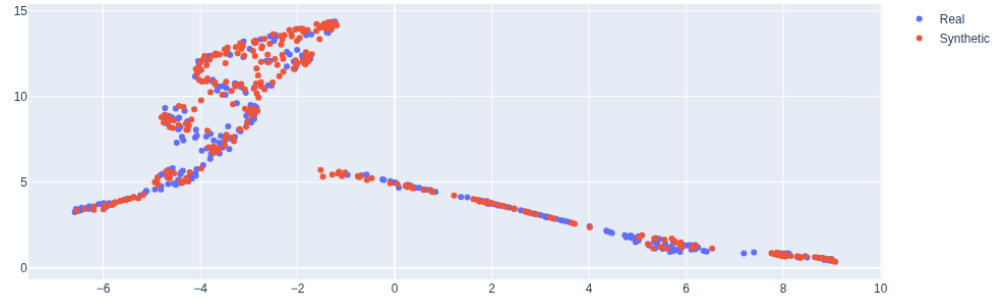


Figure 4.7: UMAP analysis of the real dataset and the TabDDPM generated dataset

	GAN dataset	TabDDPM dataset
Numerical statistical test	Rank 2	Rank 1
Categorical statistical test	Rank 1	Rank 1
Correlation matrices	Rank 1	Rank 2
Contingency tables	Rank 2	Rank 1
Final score	1.5	1.25

Table 4.3: Summary of the results in the Benchmarking section

4.4 Final score

With the Benchmarking section in SynthRO we were able to compute a final score for the two synthetic datasets, where the smallest value is a sign of a better dataset.

To compute the score, we considered the Univariate Resemblance Analysis (Kolmogorov-Smirnov test as numerical test and Chi-square test as categorical test) and Multivariate Resemblance Analysis (correlation matrices and contingency tables). Applying the same weight to all the metrics, the final results are 1.5 for the GAN generated dataset and 1.25 for the TabDDPM generated dataset. This indicates a better performance for the TabDDPM dataset, as expected.

We recall that this final score takes into consideration only the resemblance metric.

When computing the final score, we could also check the ranks of the two datasets in each one of the four metrics.

A summary is shown in Table 4.3.

Chapter 5
Discussion and conclusion

We explored the problem of studying the effects of the Rituximab treatment in patients affected by Nodular Lymphocyte Predominant Hodgkin Lymphoma (NLPHL). The difficulty arises from the fact that the available patients datasets are small, due to the small number of people affected by the disease. In our work, we analyzed the possibility of addressing this question and solving the issue of data scarcity with the help of synthetic generated data.

We implemented two different models: a specific version of a Generative Adversarial Network (HealthGAN), and a Tabular Denoising Diffusion Probabilistic Model (TabDDPM). The goal was to create an entire new dataset composed of information collected from imaginary patients affected by NLPHL, that resembles a real dataset. The real dataset that we used as reference was collected by the IRCCS Policlinico San Matteo in Pavia and was composed by data about 293 patients, half of which was treated with Rituximab. WGAN-GP and TabDDPM generated a new dataset of 1000 patients.

The HealthGAN procedure resulted in mode collapse, a very common failure of GANs. The new dataset did not resemble the original one; the diversity of the data was not captured and then not reproduced. The result was not an appropriate simulation of synthetic patients.

The Diffusion Model performed better in recreating the general aspect of the dataset, keeping the same overall statistical features. A problem was encountered when splitting the synthetic dataset into patients treated with Rituximab and non-treated patients. Confronting the survival curves, the two subgroups with the original subgroups from the real dataset, the corresponding sections behaved in opposite ways: a higher survival rate was expected from the Rituximab group, while we obtained the contrary. Considering that the main interest in clinicians' studies was to analyze the efficacy of Rituximab, that was a problem that cannot be ignored. The reason why neither of the two models performed well was attributed to the real dataset being extremely small.

Given these results obtained by the two models, neither of them was considered acceptable to be sent to the hospital.

Going back to the original question, i. e., can synthetic data help with the study of the efficacy of Rituximab, solving the problem of data scarcity?

In our study, the answer is not completely elucidated. It looks like there is potential to obtain a dataset that can replace the real smaller one when conducting a study based on patients features, and we believe the main complication in realizing this arises from the requisite of having a minimum size of the original dataset.

Comments and future works

The expectations of this study were to provide a suitable dataset to doctors, in order for them to to learn more about the efficacy of Rituximab.

In anticipation of future works, we would like to remind what are the possible improvements of our study.

In Chapter 4 we analyzed the three metrics of evaluation of a synthetic dataset: resemblance, utility, privacy. While we focus on resemblance and utility, the privacy metric is up to discussion. SynthRO is a valid method for privacy evaluation, that can be implemented before drawing conclusions on an accurate-looking dataset. This is fundamental in order to share the generated dataset.

For the utility metrics, we created survival curves to observe the number of patients without the disease progression at every step in time. For a more complete investigation, a Cox Regression [1] analysis will be able to show a multivariate inspection of the features that contribute to the disease progression.

Another aspect to consider for the evaluation of the dataset is the division between train and test dataset. As discussed in Chapter 4 we were not able to perform this division due to the small size of the dataset. Future studies with a bigger starting dataset will need this division to obtain more reliable results.

An idea for the future is to generate a synthetic dataset of a bigger size than what is needed and then introduce a dataset distillation learning [36]. This will be particularly useful when providing the new dataset to clinicians, because the final dataset will contain only the essential information. This will allow a more accurate study on the effect of interest.

Finally, our suggestion for this specific study in survival analysis is to generate a synthetic dataset without the *TIME* feature, and to train an additional predictive model on the original dataset, that will then predict the *TIME* feature for the synthetic dataset. This approach is presented in paper [26].

References

[1] Samar Abd ElHafeez, Graziella D'Arrigo, Daniela Leonardis, Maria Fusaro, Giovanni Tripepi, and Stefanos Roumeliotis. Methods to analyze time-to-event data: the cox regression analysis. *Oxidative medicine and cellular longevity*, 2021(1):1302811, 2021. (Cited on page 40)

[2] Chansik An, Yae Won Park, Sung Soo Ahn, Kyunghwa Han, Hwiyoung Kim, and Seung-Koo Lee. Radiomics machine learning study with a small sample size: Single random training-test set split may lead to unreliable results. *PLOS ONE*, 2021. (Cited on pages 11 and 30)

[3] Ashish Arora, Niloufar Shoeibi, Vishwani Sati, Alfonso González-Briones, Pablo Chamoso, and Emilio Corchado. Data augmentation using gaussian mixture model on csv files. In *International Symposium on Distributed Computing and Artificial Intelligence*, pages 258–265. Springer, 2020. (Cited on page 18)

[4] Michael Sargent Binkley, Jamie E Flerlage, Kerry J Savage, Saad Akhtar, Raphael Steiner, Xiao-Yin Zhang, Michael Dickinson, Anca Prica, Ajay Major, Peter G Hendrickson, et al. International prognostic score for nodular lymphocyte–predominant hodgkin lymphoma. *Journal of Clinical Oncology*, 42(19):2271–2280, 2024. (Cited on page 3)

[5] J Martin Bland and Douglas G Altman. Survival probabilities (the kaplan-meier method). *Bmj*, 317(7172):1572–1580, 1998. (Cited on page 29)

[6] J Martin Bland and Douglas G Altman. The logrank test. *British medical journal*, 2004. (Cited on page 29)

[7] Sven Borchmann, Erel Joffe, Craig H Moskowitz, Andrew D Zelenetz, Ariela Noy, Carol S Portlock, John F Gerecitano, Connie L Batlevi, Philip C Caron, Pamela Drullinsky, et al. Active surveillance for nodular lymphocyte-predominant hodgkin lymphoma. *Blood, The Journal of the American Society of Hematology*, 133(20):2121–2129, 2019. (Cited on page 3)

[8] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002. (Cited on page 18)

[9] Adolfo Crespo Márquez. The curse of dimensionality. In *Digital maintenance management: guiding digital transformation in maintenance*, pages 67–86. Springer, 2022. (Cited on page 10)

[10] Maduranga De Silva Dilum and Guhanathan Poravi. A review on generative adversarial networks. In *2021 6th International Conference for Convergence in Technology (I2CT)*, pages 1–4. IEEE, 2021. (Cited on page 21)

[11] Robert Flynn. Survival analysis. *Journal of Clinical Nursing*, 21(19pt20):2789–2797, 2012. (Cited on pages 26, 28, and 29)

[12] Jon Gauthier et al. Conditional generative adversarial nets for convolutional face generation. *Class project for Stanford CS231N: convolutional neural networks for visual recognition, Winter semester*, 2014(5):2, 2014. (Cited on page 44)

[13] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014. (Cited on page 21)

[14] Manuel Gotti, Roberta Sciarra, Alessandro Pulsoni, Francesco Merli, Stefano Luminari, Caterina Zerbi, Livio Trentin, Alessandro Re, Chiara Rusconi, Simonetta Viviani, et al. Role of rituximab addition to first-line chemotherapy regimens in nodular lymphocyte-predominant hodgkin lymphoma: A study by fondazione italiana linfomi. *HemaSphere*, 7(4):e837, 2023. (Cited on page 3)

[15] Han Guo and Ding Lin. Dalnet: A denoising diffusion probabilistic model for high-fidelity day-ahead load forecasting. *arXiv preprint arXiv:2503.06697*, 2025. (Cited on page 23)

[16] Daniel F. Heitjan and Srabashi Basu. Distinguishing "missing at random" and "missing completely at random". *The American Statistician*, 50(3):207–213, 1996. (Cited on page 12)

[17] Mikel Hernandez, Gorka Epelde, Ane Alberdi, Rodrigo Cilla, and Debbie Rankin. Synthetic data generation for tabular health records: A systematic review. *Neurocomputing*, 493:28–45, 2022. (Cited on pages 19 and 20)

[18] Aryan Jadon and Shashank Kumar. Leveraging generative ai models for synthetic data generation in healthcare: Balancing research and privacy. In *2023 International Conference on Smart Applications, Communications and Networking (SmartNets)*, pages 1–4. IEEE, 2023. (Cited on page 21)

[19] Zhu Jun. Synthetic data generation by diffusion models. *National Science Review*, 11(8):nwae276, 2024. (Cited on page 19)

[20] Michael G Kenward and James Carpenter. Multiple imputation: current perspectives. *Statistical methods in medical research*, 16(3):199–218, 2007. (Cited on page 12)

[21] Kristian Kleinke. Multiple imputation by predictive mean matching when sample size is small. *Methodology*, 2018. (Cited on page 12)

[22] Akim Kotelnikov, Dmitry Baranchuk, Ivan Rubachev, and Artem Babenko. Tabddpm: Modelling tabular data with diffusion models. In *International conference on machine learning*, pages 17564–17579. PMLR, 2023. (Cited on pages 20, 23, and 24)

[23] Stephen W Lagakos. General right censoring and its impact on the analysis of survival data. *Biometrics*, pages 139–156, 1979. (Cited on page 12)

[24] Pedro Machado, Bruno Fernandes, and Paulo Novais. Benchmarking data augmentation techniques for tabular data. In *International Conference on Intelligent Data Engineering and Automated Learning*, pages 104–112. Springer, 2022. (Cited on page 19)

[25] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015. (Cited on page 44)

[26] Alexander Norcliffe, Bogdan Cebere, Fergus Imrie, Pietro Lio, and Mihaela van der Schaar. Survivalgan: Generating time-to-event data for survival analysis. In *International Conference on Artificial Intelligence and Statistics*, pages 10279–10304. PMLR, 2023. (Cited on page 40)

[27] Zhaoqing Pan, Weijie Yu, Xiaokai Yi, Asifullah Khan, Feng Yuan, and Yuhui Zheng. Recent progress on generative adversarial networks (gans): A survey. *IEEE access*, 7:36322–36333, 2019. (Cited on page 21)

[28] Agapios Panos and Dimitris Mavridis. Tableone: an online web application and r package for summarising and visualising data. *BMJ Mental Health*, 23(3), 2020. (Cited on page 31)

[29] Noseong Park, Mahmoud Mohammadi, Kshitij Gorde, Sushil Jajodia, Hongkyu Park, and Youngmin Kim. Data synthesis based on generative adversarial networks. *arXiv preprint arXiv:1806.03384*, 2018. (Cited on page 26)

[30] Novella Pugliese, Marco Picardi, Roberta Della Pepa, Claudia Giordano, Francesco Muriano, Aldo Leone, Giuseppe Delle Cave, Alessandro D'Ambrosio, Violetta Marafioti, Maria Gabriella Rascato, et al. Rituximab-containing risk-adapted treatment strategy in nodular lymphocyte predominant hodgkin lymphoma: 7-years follow-up. *Cancers*, 13(8):1760, 2021. (Cited on pages 3 and 4)

[31] Robin San-Roman, Eliya Nachmani, and Lior Wolf. Noise estimation for generative diffusion models. *arXiv preprint arXiv:2104.02600*, 2021. (Cited on page 19)

[32] Gabriele Santangelo, Giovanna Nicora, Riccardo Bellazzi, and Arianna Dagliati. How good is your synthetic data? synthro, a dashboard to evaluate and benchmark synthetic tabular data. *BMC Medical Informatics and Decision Making*, 25(1):89, 2025. (Cited on pages 26 and 27)

[33] Stef Van Buuren and Karin Groothuis-Oudshoorn. mice: Multivariate imputation by chained equations in r. *Journal of statistical software*, 45:1–67, 2011. (Cited on page 12)

[34] Andrew Yale, Saloni Dash, Ritik Dutta, Isabelle Guyon, Adrien Pavao, and Kristin P Bennett. Generation and evaluation of privacy preserving synthetic health data. *Neurocomputing*, 416:244–255, 2020. (Cited on pages 20 and 22)

[35] Ling Yang, Zhilong Zhang, Yang Song, Shenda Hong, Runsheng Xu, Yue Zhao, Wentao Zhang, Bin Cui, and Ming-Hsuan Yang. Diffusion models: A comprehensive survey of methods and applications. *ACM computing surveys*, 56(4):1–39, 2023. (Cited on page 23)

[36] William Yang, Ye Zhu, Zhiwei Deng, and Olga Russakovsky. What is dataset distillation learning? *arXiv preprint arXiv:2406.04284*, 2024. (Cited on page 40)

[37] Zhaoyu Zhang, Mengyan Li, and Jun Yu. On the convergence and mode collapse of gan. In *SIGGRAPH Asia 2018 Technical Briefs*, pages 1–4. 2018. (Cited on page 22)

[38] Junbo Zhao, Michael Mathieu, and Yann LeCun. Energy-based generative adversarial network. *arXiv preprint arXiv:1609.03126*, 2016. (Cited on page 44)

Chapter 6
Appendix

GANs and Diffusion Models can also be implemented via Python libraries, respectively via *Pygan* and *Pytorch*.

6.0.1 GANs via pygan

Pygan is a free and open-source Python library, whose source code can be found on GitHub at [1]. This library was created to implement GANs, Conditional GANs (cGANs), Adversarial Auto-Encoders (AAEs), Energy-based GANs (EBGANs) and Energy-based Adversial-Auto-Encoders (EBAAEa).

cGANs is an extention of classical GAN where the model can also be *conditioned* by external information. This allows the model to work with different contextual information. [12]

AAEs are a probabilistic Autoencoder based on GAN procedure, that matches the posterior with the prior distribution of the features [25]. This ensures more meaningful samples.

EBGANs are GANs where the Discriminator acts as an energy function attributing low energies to points near the main data point in space, and higher energy to farer points. The goal of the Generator is to generate samples to whose a low energy will be attributed. [38]

EBAAEa is the approach used in Pygan, coupling between AAEs and EBGANs. In particular, tra training generator is the one of AAEs and the discriminator is taken from an EBGAN.

All of this is modeled just to create images, not other type of data.

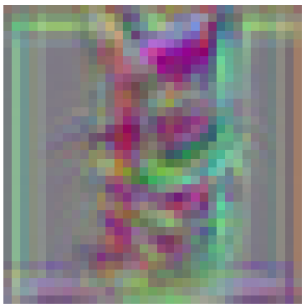


Figure 6.1: Image generated via Pygan library

[1] `https://github.com/accel-brain/accel-brain-code/tree/master/Generative-Adversarial-Networks`

You start installing the Pygan library and importing the Python module.

```
pip install pygan
from pygan._mxnet.gan_image_generator import GANImageGenerator
```

With a couple of commands you are ready to generate samples:

```
gan_image_generator = GANImageGenerator(
    # 'list' of path to your directories.
    dir_list=[
        "/path/to/your/image/files/",
    ],
    # 'int' of image width.
    width=128,
    # 'int' of image height.
    height=96,
    # 'int' of image channel.
    channel=1,
    # 'int' of batch size.
    batch_size=40,
    # 'float' of learning rate.
    learning_rate=1e-06,
    )

gan_image_generator.learn(
    # 'int' of the number of training iterations.
    iter_n=100000,
    # 'int' of the number of learning of the discriminative model.
    k_step=10,
    )
```

The result is a generated image.

For reference, Figure 6.1 represents a sample we performed to test the library. The input for the model was the Kaggle dataset [2], composed of 9997 images of cats. Due to computational time and complexity, this was generated with just 100 iterations, a number extremely small in order to obtain plausible results. This was just a test for the use of the Pygan library.

6.0.2 Diffusion Models via Pytorch

A simple and rapid implemented of diffusion models can be done in Python via the Pytorch library.

This website[3] provides a rapid implementation. This code is going to use Pytorch for building and training the neural network, NumPy for numerical operations and Matplotlib for plotting the generated images. After importing the necessary libraries, the denoising neural network can be defined.

[2] https://www.kaggle.com/datasets/crawford/cat-dataset

[3] https://www.geeksforgeeks.org/artificial-intelligence/diffusion-models-in-machine-learning/

```
class DenoisingNN(nn.Module):
    def __init__(self):
        super(DenoisingNN, self).__init__()
        self.fc = nn.Sequential(
            nn.Linear(28*28, 128),
            nn.ReLU(),
            nn.Linear(128, 28*28)
        )

    def forward(self, x):
        return self.fc(x)
```

Let's define now the two necessary processes. The forward process will add noise to the original data, while the reverse process will try to recompose the image starting from noise, taking the trained model. The first function will return the noisy data and the noise added, and the second one will return the reconstructed data.

```
def forward_process(x0, t, alpha_t):
    noise = torch.randn_like(x0)
    alpha_t = torch.tensor(alpha_t)
    xt = torch.sqrt(alpha_t) * x0 + torch.sqrt(1-alpha_t) * noise
    return xt, noise

def reverse_process(xt, model, t, alpha_t):
    xt_reconstructed = model(xt)
    return xt_reconstructed
```

At this point we need to train the model. It will iterate over the entire original dataset, applying the forward and reverse processes. It also computes the loss and update model parameters.

```
def train(model, optimizer, dataloader, num_steps=10):
    model.train()
    for step in range(num_steps):
        total_loss = 0
        for x0, _ in dataloader:
            x0 = x0.view(x0.size(0), -1)  # Flatten the images
            t = torch.tensor([0.1])    # Noise level
            alpha_t = 0.5              # Example alpha_t value

            xt, epsilon = forward_process(x0, t, alpha_t)
            optimizer.zero_grad()
            xt_reconstructed = reverse_process(xt, model, t, alpha_t)
            loss = torch.mean((xt_reconstructed -
                               x0.view(xt_reconstructed.size())) ** 2)
            loss.backward()
            optimizer.step()

            total_loss += loss.item()
            print(f"Step {step}, Loss: {loss.item()}")

        avg_loss = total_loss / len(dataloader)
        print(f"Epoch {step}, Average Loss: {avg_loss}")
```

The original dataset can be load and the model initialized. In this example, the MNIST dataset is used [4].

[4] http://yann.lecun.com

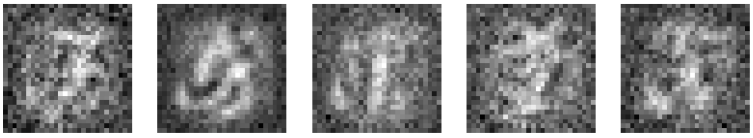


Figure 6.2: Five result examples of the diffusion model code taken from [?], trained with the MINST dataset

```
# Load the dataset
transform = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize((0.5,), (0.5,))
    ])

dataset = datasets.MNIST(root='data', train=True,
                         download=True, transform=transform)
dataloader = DataLoader(dataset, batch_size=64, shuffle=True)

# Initialize and train the model
model = DenoisingNN()
optimizer = torch.optim.Adam(model.parameters(), lr=0.001)

print("Training started...")
train(model, optimizer, dataloader, num_steps=10)
print("Training completed.")
```

To generate images, a new function *generate_images* can be created. It takes as input the model we trained in the previous step, and the number of images that we wish to generate. The output will be the set of desidered images.

```
def generate_images(model, num_images=5):
    model.eval()
    with torch.no_grad():
        noise = torch.randn(num_images, 28*28)  # Random noise
        t = torch.tensor([0.1])    # Noise level
        alpha_t = 0.5              # Example alpha_t value
        generated_images = reverse_process(noise, model, t, alpha_t)

        plt.figure(figsize=(10, 5))
        for i in range(num_images):
            plt.subplot(1, num_images, i + 1)
            plt.imshow(generated_images[i].view(28, 28).numpy(),
                       cmap='gray')
            plt.axis('off')
        plt.show()

generate_images(model)
```

Figure 6.2 shows the result obtained with steps=10, starting from the MINST dataset.