NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE

LUÍS MIGUEL DA TORRE ABREU

Bachelor's in Computer Science and Engineering

CREATION OF SYNTHETIC PATIENT DATA FOR HEALTH CARE RESEARCH

MASTER IN COMPUTER SCIENCE AND ENGINEERING

NOVA University Lisbon
September, 2025

NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

DEPARTMENT OF
COMPUTER SCIENCE

CREATION OF SYNTHETIC PATIENT DATA FOR HEALTH CARE RESEARCH

LUÍS MIGUEL DA TORRE ABREU

Bachelor's in Computer Science and Engineering

Adviser: Jörg Matthias Knorr
Associate Professor, NOVA University Lisbon

Co-adviser: João Carlos Gomes Moura Pires
Associate Professor, NOVA University Lisbon

Examination Committee

Chair: Carlos Augusto Isaac Piló Viegas Damásio
Associate Professor, Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa

Members: Carlos Soares
Associate Professor, Faculdade de Engenharia da Universidade do Porto

Jörg Matthias Knorr
Associate Professor, Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa

MASTER IN COMPUTER SCIENCE AND ENGINEERING

NOVA University Lisbon
September, 2025

Creation of Synthetic Patient Data for Health Care Research

This document was created with the (pdf/Xe/Lua)LaTeX processor and the NOVAthesis template (v7.3.9) [1].

Acknowledgements

I would like to express my gratitude to all those who were part of my learning journey.

In particular, I am deeply thankful to my supervisors for their guidance throughout these months, to my colleagues and friends who accompanied me and made this experience all the more meaningful and to the professors who contributed to the knowledge I acquired during this period.

I am also immensely grateful to my family for always being there for me and for their patience and unwavering support along the way.

”

“Not everything that can be counted counts and not everything that counts can be counted.”

— William Bruce Cameron, Informal Sociology (1963)

Abstract

In recent years, with the digitalization of the healthcare field, the use of Electronic Health Records (EHRs) has significantly increased. Their ability to store a wide range of patient information, combined with ease of access and the capacity for smooth data sharing among hospitals and medical providers, has made them an essential tool in modern healthcare. While EHRs are not yet perfect, the data they contain is highly valuable for professionals and researchers in the healthcare field. This data has the potential to drive advancements and breakthroughs in medical research. However, since EHRs contain personal and sensitive information about patients, their content must be protected, leading to challenges in protecting privacy while making it available for research purposes. Synthetic data, a field that has also been rapidly growing in recent years, primarily driven by advancements in machine learning, is a potential solution for this problem. By generating synthetic datasets that replicate real world data, it is possible to preserve the characteristics and quality of the original dataset while addressing privacy concerns. In order to protect patient privacy, synthetic data must not resemble the original dataset or allow re-identification.

The goal of this work is to explore the potential of synthetic data generation when applied to a dataset of clinical records referred to as High Users (HU). This dataset contains both demographic information about patients and a four year history of medical appointments, emergency episodes and hospitalizations. Using the High Users dataset, the objective is to create synthetic data that replicates the original dataset's characteristics and quality through machine learning techniques, while ensuring privacy, meaning that the synthetic data should not allow the re-establishment of the original dataset. The study analyzes multiple approaches for generating synthetic data that retain the properties of real world data. The experiments conducted determine whether the selected synthetic data generation techniques are suitable for the HU dataset, but the results were largely negative, with standard evaluation metrics also proving disappointing in assessing effectiveness in preserving properties while ensuring privacy.

Keywords: Synthetic Data, Healthcare, Privacy, Machine Learning, Electronic Health

Records

Resumo

Nos últimos anos, com a digitalização da área da saúde, o uso de Registos Eletrónicos de Saúde (RES) aumentou significativamente. A capacidade de armazenar um amplo conjunto de informações sobre pacientes, aliada à facilidade de acesso e à partilha eficiente de dados entre hospitais e clínicas, tornou-os uma ferramenta essencial na saúde moderna. Embora os RES ainda não sejam perfeitos, os dados que contêm são extremamente valiosos para profissionais e investigadores na área da saúde, com potencial para impulsionar avanços na investigação médica. No entanto, como os RES contêm informações sensíveis e pessoais dos pacientes, é necessário protegê-los, o que gera desafios para equilibrar a privacidade e a disponibilidade dos dados para fins de investigação.

A geração de dados sintéticos, uma área que também tem crescido rapidamente nos últimos anos, surge como uma potencial solução para esse problema. Por meio da criação de dados sintéticos que replicam dados do mundo real, é possível preservar as características e a qualidade dos dados originais, ao mesmo tempo que se abordam as questões da privacidade. Para proteger a privacidade dos pacientes, os dados sintéticos não se devem assemelhar ao conjunto dos dados original nem permitir a re-identificação.

O objetivo deste trabalho é explorar o potencial da geração de dados sintéticos ao ser aplicada a um conjunto de dados clínicos designado *"High Users"* (HU). Este conjunto inclui tanto informação demográfica dos pacientes como um histórico de quatro anos de consultas, episódios de urgência e internamentos. Utilizando o conjunto HU, o objetivo é criar dados sintéticos que mantenham as características e a qualidade do conjunto original por meio de técnicas de aprendizagem automática, garantindo a privacidade, ou seja, que os dados sintéticos não permitam a reconstituição do conjunto original. O estudo analisa múltiplas abordagens de geração de dados sintéticos que procuram preservar as propriedades dos dados reais. As experiências realizadas determinaram se as técnicas selecionadas são adequadas ao conjunto HU, mas os resultados foram maioritariamente negativos, com as métricas de avaliação padrão a revelarem-se dececionantes na análise da eficácia em preservar propriedades e garantir privacidade.

Palavras-chave: Dados Sintéticos, Saúde, Privacidade, Aprendizagem Automática, Registos Eletrónicos de Saúde

Contents

List of Figures

Glossary

encryption A security technique that transforms data into a coded or unreadable format, typically using mathematical algorithms and cryptographic keys, to prevent unauthorized access. Only authorized parties with the correct decryption key can restore the original information. *(p. 8)*

FAIR An acronym that stands for Findable, Accessible, Interoperable and Reusable. It refers to a set of guiding principles for scientific data management and stewardship, aiming to enhance the ability of machines and humans to find, access, integrate and reuse data effectively. *(pp. 7, 11, 14)*

normalization A preprocessing technique used to adjust the scale of numerical data so that values from different features become comparable. Normalization typically transforms data into a specific range, such as $[0, 1]$, or adjusts it to have unit norm. *(pp. 19, 79)*

overfitting A modeling error that occurs when a machine learning model learns not only the underlying patterns in the training data but also the noise and random fluctuations. As a result, the model performs very well on the training dataset but generalizes poorly to new, unseen data. Overfitting is often associated with excessive model complexity, such as too many parameters relative to the amount of available data. *(pp. 64, 68, 70, 72, 73, 75)*

Acronyms

EHRs	Electronic Health Records *(pp. 1–4, 8, 18, 24–28)*
ER	Entity–Relationship *(pp. xi, 34, 41, 42)*
GAN	Generative Adversarial Network *(pp. 3, 16–19, 21, 22)*
GDPR	General Data Protection Regulation *(p. 2)*
HIPAA	Health Insurance Portability and Accountability Act *(p. 2)*
HITECH	Health Information Technology for Economic and Clinical Health *(p. 2)*
HL7	Health Level Seven International *(pp. 7, 11, 14)*
HMMs	Hidden Markov Models *(pp. 25, 77)*
HU	High Users *(p. 33)*
JS Distance	Jensen-Shannon Distance *(pp. 29, 30)*
MLQS	Machine Learning Quality Score *(pp. 31, 68, 73)*
PCA	Principal Component Analysis *(p. 29)*
PII	Personally Identifiable Information *(pp. 8, 37)*
RNNs	Recurrent Neural Networks *(pp. 7, 13, 14, 23, 25)*
SDV	Synthetic Data Vault *(pp. 15, 23)*
SQS	Synthetic Quality Score *(p. 29)*

1

Introduction

This chapter is structured into four main sections: *Context and Motivation*, *Approaches to the Problem*, *Contributions* and *Document Overview*.

The first section introduces the background of the problem, outlining the importance of electronic health records and the privacy challenges associated with their use. The second section presents existing strategies for generating synthetic data, contrasting statistical and machine learning–based methodologies. The third section details the main objectives and contributions of this thesis. Finally, the *Document Overview* provides an overview of the overall thesis structure.

1.1 Context and Motivation

Machine learning has become a vital resource in today's society, easily integrating into diverse fields of work. Its significance has grown to the extent that many people consider it an essential tool. With this trend, machine learning has also gained prominence in the domain of data analysis. As data continues to grow in volume and importance, it serves as a significant resource for acquiring deeper insights and has become a fundamental component of decision making and innovation across sectors [2].

Among the many sectors transformed by machine learning, healthcare stands out for its potential to significantly enhance quality of care, drive research advancements and improve operational efficiency. Machine learning is already being utilized in healthcare to enhance diagnostic accuracy and predict patient risks. However, its effectiveness heavily depends on the quality and availability of the data it relies on. High-quality, accessible data is essential for ensuring that models produce reliable results that can be trusted in clinical settings [3].

Electronic Health Records (EHRs) are an example of data that has become essential in modern healthcare. EHRs are digital versions of patients' paper charts, offering patient-centered records that are accessible to authorized healthcare providers. The utilization of EHRs provides numerous advantages, such as improved patient care, greater efficiency in information sharing between clinical entities, higher data accuracy and the ability

to store a wide range of patient information. Building on this, EHRs not only capture demographic and administrative details but also encompass a diverse range of clinical data, including medical histories, diagnoses, laboratory test results, prescriptions, vital signs and procedural reports. This information is characterized by logical dependencies, as different data points influence and build upon one another over time. For instance, a diagnosis may lead to specific laboratory tests, which in turn impact treatment decisions and follow-up care. As a result, EHRs reflect the interdependence of clinical events over time, where records are continuously updated with new medical appointments, hospital admissions and treatments. This temporal dimension is crucial for tracking disease progression, identifying clinical patterns and enabling personalized patient care. However, as compelling as the advantages may be, the growing utility and importance of EHRs bring with them increasing concerns about privacy [4, 5].

EHRs contain sensitive personal information that must be protected from unauthorized disclosure. Every citizen has the right to the privacy of their clinical data and is the only individual with legitimate authority to grant permission for temporary access and sharing of this information. The way in which this access is granted depends on the purpose for which the data is used. In a clinical setting, EHRs are accessed on a case by case basis, with healthcare professionals retrieving the necessary patient data to support diagnosis, treatment and follow up care. This access is inherently limited and temporary, ensuring that only authorized individuals directly involved in a patient's care can view the relevant information. Beyond individual care, EHRs also serve as an invaluable resource for medical research and clinical investigations. In this context, data is typically accessed in aggregate form, often spanning large patient populations and extended time periods. Unlike clinical access, this type of usage is not intended for direct patient treatment but rather to identify trends and advance medical knowledge. Given the broader scope and prolonged duration of access, strict safeguards such as privacy-preserving techniques are essential to protect patient confidentiality. In the United States, the Health Insurance Portability and Accountability Act (HIPAA) and the Health Information Technology for Economic and Clinical Health (HITECH) establish rigorous standards for safeguarding electronic health information, while in Europe, the General Data Protection Regulation (GDPR) enforces data privacy and security measures. These restrictions limit access and make it challenging to fully leverage the potential of EHRs, despite the highly valuable data they hold for healthcare professionals and researchers [4, 5]. However, no matter how beneficial digital data may be for achieving scientific progress, clinical organizations can only truly capitalize on its potential if total security and reliability in protecting patients' sensitive information is guaranteed.

One of the earliest solutions proposed to address this issue was the use of anonymized data to mitigate privacy concerns. However, studies have demonstrated that anonymized datasets can sometimes be re-identified through advanced techniques, thereby compromising the privacy of individuals [6, 7]. For instance, even if direct identifiers such as

names and social security numbers are removed, an attacker could still re-identify individuals by cross-referencing seemingly innocuous attributes. Suppose a dataset includes rare medical conditions, age and ZIP codes by linking this information with publicly available records (e.g., voter registration databases or social media posts mentioning a diagnosis), an individual's identity could be inferred. This showcases the need for more robust methods to protect data privacy while preserving its utility.

Synthetic data, that is rapidly growing across sectors driven by its ability to solve various challenges, presents itself as a possible solution to this problem as it can replicate the characteristics and quality of real world data while addressing privacy concerns. To safeguard patient privacy, it is vital that synthetic data neither directly mirrors the original dataset nor permits re-identification.

In addition, synthetic data provides a solution to data scarcity, as it can be generated to augment existing datasets or simulate scenarios that may be rare or difficult to collect data on [8].

1.2 Approaches to the Problem

1.2.1 Methodologies

Among the most relevant and popular strategies for synthetic data generation are statistical methods and machine learning based approaches:

- **Statistical Methods**: Tools like *Synthea* [9] use predefined statistical models and publicly available data to create synthetic datasets that reflect population-level trends, such as disease prevalence, treatment frequencies and age-related health conditions as well as demographic consistency, ensuring realistic distributions of age, gender and ethnicity. While these approaches are useful for creating generalized data, they are limited by predefined rules and therefore often fail to capture the complex, emergent relationships and nuanced temporal dependencies observed in real Electronic Health Records.

- **Machine Learning-Based Methods**: Generative Adversarial Network (GAN) [10] and autoregressive models [11] have proven to be highly effective in generating synthetic data that preserves the underlying structure of real data, including statistical distributions, correlations between variables and temporal patterns. Frameworks such as *medGAN* [12] and *CTGAN* [13] perform well in generating high quality tabular data, while advanced models like *DoppelGANger* [14] and *ParSynthesizer* [15] aim to address sequential and temporal dependencies, making them more suitable for EHRs data. Moreover, *DP-ACTGAN* [16] was developed to address the privacy concerns associated with traditional GAN, particularly their vulnerability to leaking sensitive information from the training dataset.

1.2.2 Promising Methodologies

Given the sequential and structured nature of EHR data, it is important to account for their specific properties, such as logical dependencies and the interdependence of clinical events over time. The following methodologies have the ability to address these complexities and, consequently, are considered promising for maximizing the potential of EHRs data:

- **DoppelGANger**: This model effectively separates static and dynamic features, which makes it well-suited for modeling sequential data as its capacity to capture temporal dependencies aligns closely with the requirements of EHRs data generation.
- **ParSynthesizer**: With its probabilistic and autoregressive framework, ParSynthesizer provides a robust approach for generating sequential data.
- **ACTGAN**: Although primarily designed for tabular data with logical dependencies, ACTGAN is explored with necessary modifications to incorporate time series modeling techniques. This enables the model to address the temporal dependencies inherent in EHRs data while maintaining logical consistency between attributes.

This thesis explores the use of synthetic data as a potential solution to address privacy restrictions, the difficulty of preserving temporal and logical dependencies and the limitations of existing anonymization or statistical approaches, through a case study that evaluates its applicability within the healthcare context. The challenge proposed by *VOH.CoLAB* [17] involves applying synthetic data generation to the clinical records provided by Hospital Garcia da Orta. This study focuses on assessing the effectiveness of the selected methodologies in creating a synthetic replica of the original dataset, resulting in a new artificial set of records that are adequate for use in research.

The new synthetic dataset would eliminate privacy concerns, as it would ensure that the original data remains uncompromised. It would also maintain the same characteristics as a real dataset. This way, the resulting synthetic clinical records could be used similarly to the synthetic EHRs described previously, allowing us to fully reap their benefits.

.

1.3 Contributions

This study addresses multiple challenges in the field of synthetic data generation by exploring and applying advanced techniques designed to create high quality synthetic data. It also aims to provide new valuable knowledge and practical solutions by generating synthetic data that accurately replicates the structure of the patient records, enabling the evaluation of the effectiveness of various methodologies.

The contributions of this work covers both theoretical analysis and practical implementation. The following points outline the principal contributions of this work:

- **Generation and Benchmarking of Synthetic Data:** This work aims to fulfill the objectives set by *VOH.CoLAB* by generating synthetic data derived from patient records, ensuring that the output both reflects the structure of the original dataset and preserves patient privacy.
  To achieve this, three main representative generative approaches were implemented and systematically benchmarked - `PARSynthesizer`, `DoppelGANger` and `ACTGAN`. Their capacity to capture statistical distributions, reproduce structural dependencies and model temporal sequences was assessed, revealing both their strengths and limitations.

- **Discussion of Synthetic Data Techniques**: Present a detailed analysis of the most effective techniques for generating synthetic data, particularly in the context of sequential data (the type of data used in the study), emphasizing the advantages, limitations and suitability of each method.

- **Development of an Evaluation Pipeline:** An evaluation framework was designed and implemented to overcome the shortcomings of standard integrity, utility and privacy metrics. This pipeline includes manual and rule-based validation steps (e.g., chronological consistency checks, enforcement of constraints,) that complement automated metrics. By combining statistical assessment with domain-specific plausibility tests, the pipeline provides a more accurate picture of the usefulness and reliability of synthetic data in healthcare contexts.

- **Addressing Challenges of Real Data Usage:** Provide insights and propose solutions to overcome the limitations and challenges imposed by the use of real world data, focusing on privacy, security and ethical constraints.

1.4 Document Overview

This document[1] is organized into six chapters:

1. **Introduction**: The first chapter introduces the problem, its context and the motivation for addressing it. It also presents some of the methodologies that can be applied to solve the problem and outlines the contributions expected from this work.

2. **Synthetic Data - Background and Research**: This chapter begins by clarifying crucial concepts essential for understanding the detailed analysis presented later. It examines previous work and methodologies that have addressed similar problems, emphasizing their strengths, limitations and relevance to this study.

[1] Generative AI tools were used for text refinement and grammatical review. However, their role was limited to assisting in language clarity and consistency and the author assumes full responsibility for the content presented.

3. **Data Analysis**: This chapter focuses on analyzing the data provided for this study. It provides a detailed overview of the dataset, examines its structure and identifies main issues, laying the groundwork for the synthetic data generation methodologies to be applied.

4. **Methodologies**: This chapter presents the different synthetic data generation models applied in this study. It explains the configuration of each model, the preprocessing steps involved and the necessary transformations required to align the clinical dataset with each model's input constraints. The chapter also discusses challenges related to data volume, null handling and categorical encoding, along with structural limitations such as the lack of support for multi-table relationships or logical constraints.

5. **Evaluation of Results**: This chapter evaluates the performance of each methodology using a set of metrics. The results are compared across multiple dimensions, such as integrity, privacy and utility, providing insights into the strengths and weaknesses of each approach.

6. **Conclusion and Future Work**: The final chapter summarizes the main findings of the study, reflects on the challenges encountered and discusses potential improvements. It also outlines directions for future research in the field of synthetic data generation.

2

Synthetic Data: Background and Research

This chapter is organized into four sections: *Related Concepts*, *State of the Art*, *Discussion* and *Evaluation Metrics*. This structure is designed to first introduce and clarify fundamental concepts in *Related Concepts*, which will serve as a foundation for understanding the methodologies discussed in *State of the Art*. This ensures that the necessary context is provided to grasp the detailed information about existing research in the field. Next, the *Discussion* serves to summarize what has been learned from the *State of the Art*, emphasizing the most significant insights. Finally, the *Evaluation Metrics* section outlines the approaches for assessing and comparing synthetic data generation techniques.

2.1 Related Concepts

In this section, it is explained key concepts that are essential for understanding the subsequent technical discussions in *State of the Art*.

The topics covered include an overview of Electronic Health Records and their importance in healthcare data management, the definition and challenges of handling sensitive data and the risks and limitations associated with anonymization techniques. As synthetic data is one of the most central topics of this work, we define it thoroughly and explore its use cases across various domains, with a particular focus on applications in healthcare [18]. Additionally, we discuss healthcare data standards, such as Health Level Seven International (HL7) and the FAIR principles, which have a crucial role in enhancing data usability in research. Building on this foundation, we delve into the nature of the data itself, distinguishing between discrete and continuous data types to better understand their characteristics.

Finally, we address important technical challenges, including mode collapse in generative models, dependencies in sequential data and the use of Recurrent Neural Networks (RNNs) for handling sequential data.

2.1.1 Electronic Health Information

Eletronic Health Records

Electronic Health Records are digital systems that store extensive and continuous records of patient healthcare data. These records include demographic information, medical history, prescriptions and laboratory results [19]. While EHRs enhance healthcare efficiency by enabling better data access and sharing, they also present challenges in standardization and privacy protection. Different institutions use varying formats, hindering interoperability, while the sensitive nature of the data necessitates strong security measures [5].

Sensitive Data

Healthcare data is inherently sensitive, containing personally identifiable and medical information that, if exposed, could lead to identity theft, discrimination or misuse. Due to the potential consequences of unauthorized access or misuse, handling sensitive healthcare data requires strict adherence to regulations such as the General Data Protection Regulation in Europe and the Health Insurance Portability and Accountability Act in the United States, ensuring compliance with ethical and legal standards [20]. The sensitivity of this information in the healthcare context emphasizes the importance of secure data management techniques, including encryption, anonymization and the use of synthetic data.

Anonymization and Privacy Risks

Privacy is a fundamental principle in healthcare data management, ensuring that sensitive patient information is protected from unauthorized access, misuse or breaches. In the context of EHRs, privacy involves safeguarding Personally Identifiable Information (PII) while enabling the secure and ethical use of data for research and innovation.

Privacy protection methods include encryption, access controls and secure sharing protocols. However, even with these measures, ensuring privacy often requires transforming the data itself [5]. A common approach to achieve this is anonymization.

Anonymization involves removing or obfuscating personally identifiable information from datasets so that individuals cannot be directly identified. However, anonymization is not infallible. Re-identification attacks, where anonymized data is cross-referenced with external datasets, can still compromise privacy. For example, in the Netflix Prize dataset, researchers were able to re-identify individuals by correlating their movie ratings with publicly available reviews on IMDb, demonstrating how seemingly harmless data can still reveal personal identities [21]. This emphasizes the need for more robust privacy-preserving techniques, such as synthetic data generation, to mitigate these risks effectively [6].

In the healthcare domain, where sensitive information is at stake, the balance between data utility and privacy remains a complex and permanent challenge [16].

Metadata

Metadata refers to structured information that describes, explains, locates, or otherwise makes it easier to retrieve, use, or manage data. In the context of relational databases and synthetic data generation, metadata includes schema definitions, table relationships, primary and foreign key constraints, data types and other structural rules that govern the organization and interpretation of data [22]. The impact that metadata has on the quality of the associated datasets is significant and direct. The more comprehensive and well documented the metadata, the greater the assurance of data usability over time, allowing any team member to comfortably explore, reuse and reproduce analyses.

2.1.2 Differential Privacy

Differential privacy ensures that an individual's presence or absence in a dataset minimally impacts query results [23]. This is achieved by adding controlled noise to the data or its outputs, limiting the risk of re-identification. A mechanism satisfies ε-differential privacy if the probability of obtaining a result from two datasets differing by one record remains nearly identical:

$$P(M(D) = S) \leq e^{\varepsilon} P(M(D') = S)$$

In this definition, D and D' represent neighboring datasets that differ by only a single individual record, while M denotes the randomized mechanism applied to the dataset. The expression $M(D)$ refers to the output of M when executed on D and S corresponds to a possible subset of outcomes. The probabilities $P(M(D) = S)$ and $P(M(D') = S)$ therefore indicate the likelihood of obtaining the same result when the mechanism is applied to either dataset. The parameter ε controls the trade-off between privacy and accuracy: smaller values enforce stronger privacy by ensuring that the two probabilities remain very close, whereas larger values allow more deviation in favor of accuracy. The exponential term e^{ε} quantifies the maximum difference permitted between the outputs of D and D', thereby guaranteeing that the presence or absence of a single individual does not significantly influence the result. Widely used in applications such as Apple's data collection and Google's analytics, differential privacy also strengthens synthetic data models, such as DP-ACTGAN, for healthcare data protection [16].

2.1.3 Discrete and Continuous Data

Having addressed the risks associated with protecting sensitive data, it is important to examine the nature of the data itself. In healthcare, datasets often contain a mix of discrete and continuous data types, each with unique characteristics and challenges [15].

Discrete data consists of distinct, countable values and can be further categorized into non-ordinal and ordinal types. Non-ordinal discrete data includes categorical variables with no inherent ranking, such as blood type (A, B, AB, O) or diagnostic codes, which can have either low or high cardinality depending on the number of possible values.

Ordinal discrete data, on the other hand, represents variables with a meaningful order but without precise numerical differences between levels, such as pain severity (mild, moderate, severe) or cancer staging (Stage I, II, III, IV). Numerical discrete data can also be classified based on the scale of measurement. In interval scales, the differences between values are meaningful, but there is no true zero point, such as temperature in Celsius or Fahrenheit. In contrast, ratio scales have a meaningful zero point, allowing for comparisons based on absolute magnitude, such as the number of patient visits, heart rate or medication doses.

Meanwhile, continuous data represents measurements that can take any value within a given range and are typically collected from medical tests or monitoring devices. Examples include a patient's blood pressure, body temperature, glucose levels or oxygen saturation, which vary across a spectrum rather than falling into discrete categories [24].

2.1.4 Synthetic Data

A promising solution to the challenges of handling sensitive healthcare data, while ensuring its usability for research and innovation, lies in the generation of synthetic data. Synthetic data refers to artificially generated datasets that replicate the properties of real world data, such as statistical distributions and temporal patterns, ensuring that the synthetic dataset retains the same characteristics and utility as the original data while preserving privacy [25].

There are different types of synthetic data [8, 18]:

- **Fully Synthetic Data**: Entirely generated from scratch based on models trained on real datasets. This type ensures maximum privacy but may lack specific nuances of the original data because the generation process relies on learned patterns and approximations rather than direct observations. As a result, subtle variations or complex dependencies that exist in real world data may not be fully captured [18].
- **Partially Synthetic Data**: Combines synthetic records with real data, typically replacing only the most sensitive parts of the dataset while keeping other features unchanged. For example, in a clinical dataset, patient identifiers and rare diagnoses might be synthetically generated, while general medical history and laboratory results remain real [18].
- **Hybrid Synthetic Data**: Merges synthetic data with anonymized real data, aiming to balance privacy and realism. Unlike partially synthetic data, which replaces only selected attributes, hybrid synthetic data integrates anonymized real records with fully synthetic records. For instance, a hospital might use real patient demographics with anonymization but generate synthetic medical histories to protect sensitive details while maintaining overall data realism [18].

As outlined by James [8] on synthetic data use cases, synthetic data and its applications can be explored across a wide range of areas, as represented in the Figure 2.1. In the

healthcare context, synthetic data provides a secure yet effective alternative to real data, acting as a key enabler of innovation, especially when aligned with data standards like HL7 and the FAIR principles, which will be explored in more detail later.

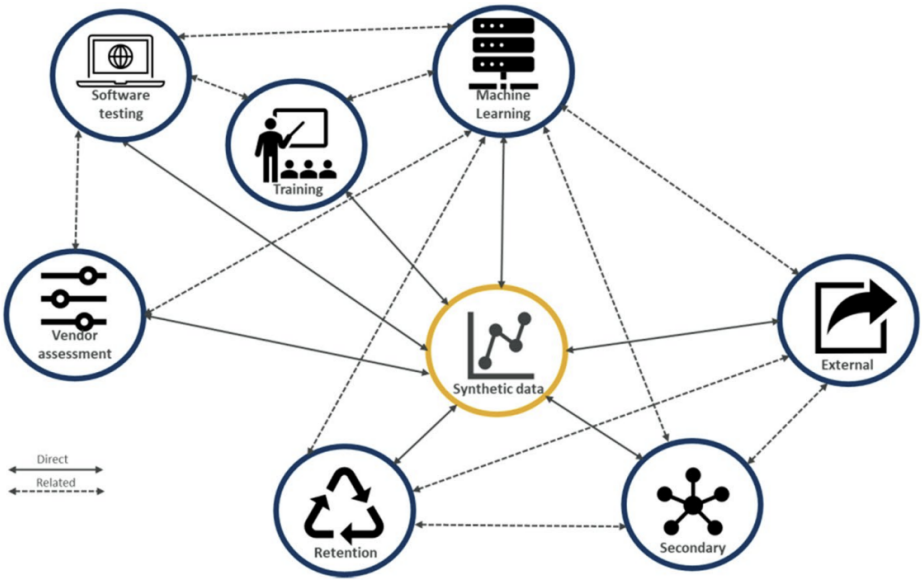


Figure 2.1: Examples of use cases for synthetic data [8].

Some examples of use cases for synthetic data in healthcare include:

- Training and validating machine learning models in environments where real data is inaccessible.
- Sharing data between institutions for collaborative research while maintaining strict privacy compliance.
- Accelerating innovation and research by providing high quality, privacy preserving datasets that enable the development of new healthcare technologies and solutions. One example is *Synthea™*. It creates realistic, lifelike Electronic Health Records for research and testing purposes without exposing real patient data [26].

2.1.5 Modeling Sequential Data: Characteristics and Dependencies

Sequential data can be modeled using a tabular format, which often involves handling sequences and multi-sequences [15].

- **Sequence**: A sequence represents a single ordered set of events or observations associated with a unique identifier, such as the data recorded for a single patient over time. It may optionally include an additional separate column, called sequence index, such as a timestamp, that reinforces that order.

- **Multi-Sequence**: A multi-sequence dataset contains multiple sequences, each corresponding to a different entity, such as multiple patients, where each sequence is identified by a unique key (e.g., Patient ID).

Real world datasets often contain multiple sequences within a single table. In such multi-sequence datasets, different sequences exist independently of each other but share the same tabular structure. Only the rows that belong to the same sequence exhibit inter-row dependencies, as their values are influenced by the temporal or contextual relationships within the sequence itself. As illustrated in Figure 2.2, multi-sequence data includes both static attributes (e.g., gender or smoker status) and time-variant attributes (e.g., heart rate) for each sequence. To create synthetic sequential data for real world datasets, it is important to consider the different properties that the sequential data may have [15]:

- A mix of data types, including numeric, categorical, datetime, etc., along with the possibility of missing values.
- The possibility that multiple sequences can be present in the same table and can have different lengths.
- The possibility that some context data remains constant throughout the sequence, while others may change occasionally. For example, birth date is a fixed attribute, whereas smoking status may change over time.

	Patient ID	Sex	Smoker	Time	Heart Rate	Systolic BP	Diastolic BP
$S_0^{(0)}$	ID_000	M	True	01/02/2020	ELEVATED	110	82
$S_1^{(0)}$	ID_000	M	True	01/14/2020	ELEVATED	108	NaN
$S_2^{(0)}$	ID_000	M	True	01/24/2020	RESTING	119	88
...	...	...	...	...	...	...	...
$S_0^{(1)}$	ID_001	F	False	01/03/2020	RESTING	120	80
$S_1^{(1)}$	ID_001	F	False	01/23/2020	Missing	118	79
$S_2^{(1)}$	...	...	...	...	...	...	...

Figure 2.2: Example of multi-sequence data that contains different sequences as defined by the sequence key, Patient ID [15].

When working with sequential data, it is common to encounter dependencies between rows. These dependencies can be broadly categorized into **inter-row dependencies** and **logical dependencies** [27]. An inter-row dependency occurs when the data or values in one row of a dataset are influenced by the data or values in another row. In healthcare, such dependencies are particularly prevalent, for example, a patient's blood pressure reading today may be influenced by their medication intake yesterday. These inter-row dependencies often take the form of temporal dependencies, reflecting the sequential

nature of events or observations over time. In contrast, logical dependencies refer to relationships or constraints among elements in the data that must hold true based on logical reasoning or domain specific rules. Unlike inter-row dependencies, these are often abstract and not directly tied to time or sequence. Logical dependencies are crucial to maintain the integrity, coherence and functionality of a system. In healthcare, logical dependencies can be observed in treatment pathways, where a prescribed medication depends on the patient's diagnosis, which in turn is based on recorded symptoms and laboratory test results. For instance, a diagnosis of diabetes may lead to blood glucose tests, which then determine the need for insulin or other medications.

To effectively model inter-row and logical dependencies in sequential data, the datasets should be structured using a tabular format that captures important identifiers, contextual information and time variant data, as illustrated in Figure 2.2:

- **Sequence Key**: A unique identifier, such as the patient ID in the Figure 2.2, to track the sequence of events for each patient.
- **Contextual Information**: Static patient information, such as their name or gender, that does not change over time, it remains constant across all rows for the same sequence key.
- **Time-Variant Data**: Dynamic attributes that change with each event, they vary across rows for the same sequence key, such as laboratory results, vital signs or diagnoses.

2.1.6 Mode Collapse

After understanding the nature of the data, it is essential to address the challenges inherent in generating synthetic data. A prominent issue is mode collapse, a frequent problem in generative models that significantly impacts the diversity and quality of the generated data.

This phenomenon, which will be explored further in the *State of the Art* section, occurs when the generator fails to capture the full diversity of the data distribution, resulting in outputs with limited variations. For instance, in the context of healthcare data generation, mode collapse might result in synthetic data that overly resembles a small subset of the original dataset, thereby reducing its utility and compromising generalizability [10, 13].

Addressing mode collapse is particularly important in the context of sequential data, where inter-row dependencies play a crucial role in maintaining temporal and logical relationships.

2.1.7 Recurrent Neural Networks

To address the sequential nature of healthcare data, advanced techniques like Recurrent Neural Networks are often used. RNNs are a class of neural networks designed to process

sequential data. Unlike traditional feedforward networks, RNNs include connections that form directed cycles, enabling them to retain information from previous inputs [14].

This makes them the ones to use for tasks such as time series forecasting, natural language processing and, the most relevant for this work, analyzing patient progression over time. Despite their strengths, RNNs are prone to challenges such as vanishing and exploding gradients, which can limit their performance on long sequences. These challenges arise during the training process, when the network adjusts its weights using backpropagation through time (BPTT).

The issue of vanishing gradients occurs when the gradients of the loss function with respect to earlier layers become exceedingly small as they are propagated backwards through the network. As a result, the weights of these earlier layers receive very small updates, causing the network to struggle to learn long-term dependencies. This leads to the network favoring recent inputs over earlier ones in the sequence.

On the other hand, exploding gradients occur when gradients become excessively large during backpropagation, often causing network weights to grow exponentially, leading to instability in the training process. Exploding gradients can cause the model's weights to diverge, resulting in high loss values and preventing the model from converging [28].

2.1.8 HealthCare Data Standards

In healthcare, data management and exchange rely on established standards to ensure compatibility, interoperability and usability across systems. Standards like HL7 [29] define rules and formats for secure and efficient data exchange, enabling seamless communication between platforms and institutions. Additionally, the FAIR principles provide guidelines for improving data findability, accessibility, interoperability and reusability in research [30].

However, in this study, the dataset consists of CSV files that do not follow these structured formats. Therefore, an extensive exploration of healthcare data standards falls outside the scope of this study.

2.2 State of the Art

This section explores the diverse methodologies developed to tackle the challenge of generating synthetic data while preserving privacy and ensuring data utility, ranging from statistical approaches to sophisticated machine learning frameworks.

Early approaches, such as data perturbation techniques, focused on directly modifying real datasets by introducing noise, masking attributes, or applying differential privacy mechanisms [31]. While these methods provided foundational privacy protection, they often struggle with maintaining data utility, particularly in complex datasets with interdependencies.

Statistical and simulation-based approaches, like *Synthea*, focus on generating synthetic data at the population level [9]. To enhance privacy, *Synthea* incorporates *PaDARser*, which anonymizes public datasets while maintaining the overall statistical properties and real-world data distributions. However, their reliance on predefined rules and publicly available statistics often limits their ability to capture individual level variability or complex temporal dependencies [9].

In contrast, modern neural network-based methods, such as *Generative Adversarial Networks* and *Autoregressive models*, adopt a data-driven approach to synthetic data generation [10]. Autoregressive models and tools like *ParSynthesizer*, specialize in modeling sequential data [11]. ParSynthesizer, developed within the Synthetic Data Vault (SDV) framework [15], extends traditional autoregressive principles to support multi-sequential datasets [32]. It captures temporal dependencies while maintaining statistical fidelity to the original dataset, making it particularly effective for generating synthetic electronic health records, simulating disease progression and modeling treatment outcomes.

Recent approaches have also explored probabilistic models such as Markov Chains [33], which define state transitions based solely on the current state and multi-table synthesizers [34], which aim to preserve relational structure and foreign key dependencies across interconnected tables.

2.2.1 Data Perturbation Techniques

Before the advent of generative models, data perturbation techniques were widely used to enhance data privacy while maintaining usability. These methods modify real data by introducing randomness or altering specific features to reduce the risk of re-identification. While simpler than modern generative models, perturbation remains a foundational approach, particularly in sensitive domains like healthcare [31]. Some key perturbation techniques include:

- **Noise Addition**: Introduces random noise to continuous data to obscure exact values while maintaining overall trends. Common approaches include Gaussian and Laplacian noise. For example, in a hospital dataset containing patient ages, adding Gaussian noise could modify an age of 65 to 63.7 or 66.5, ensuring privacy while preserving the general age distribution in the dataset.

- **Feature Masking**: Obscures or removes specific attributes, such as names or unique identifiers, to prevent direct identification.

- **Attribute Swapping**: Randomly reassigns sensitive attributes across records, breaking direct links between individuals and their data.

- **Differential Privacy**: Adds calibrated noise to datasets or query outputs to ensure minimal impact from any single data point.

Although data perturbation is computationally efficient and reduces re-identification risks, it has significant drawbacks. Higher noise levels degrade data utility and fail to preserve complex dependencies, limiting its use in high-fidelity applications. Moreover, advanced re-identification techniques can still exploit perturbed data by cross-referencing external sources [35]. For example, even if birthdates in a medical dataset are slightly modified using noise, attackers could match them with public voter records to re-identify individuals. Due to these limitations, modern privacy-preserving approaches increasingly rely on machine learning techniques, such as *Generative Adversarial Networks* and autoregressive models, to generate high-fidelity synthetic data while preserving privacy.

2.2.2 Generative Adversarial Networks

Building on the limitations of data perturbation techniques and traditional statistical approaches, Generative Adversarial Networks represent a significant advancement in synthetic data generation. Unlike earlier methods that struggled to balance privacy and data utility, GANs leverage deep learning to model complex patterns and distributions in data.

In recent years, GANs have achieved significant success in the field of data generation, demonstrating remarkable capabilities in producing high quality synthetic images [10, 16]. Within the domain of medical imaging, GANs have gained popularity for their ability to generate realistic images, such as X-rays and MRI scans, which can be utilized for data sharing and model training. This approach enables researchers to access valuable datasets while safeguarding patient privacy by eliminating the need to disclose information from real patients [16].

The GAN methodology consists of two neural networks trained simultaneously in a framework (Figure 2.3):

- **Generator**: Attempts to generate realistic, but fake data with the goal of mimicking real samples.
- **Discriminator**: Aims to distinguish between the fake data generated by the generator and the real data.

Through this adversarial process, the generator keeps improving its ability to create realistic data [10].

The training process uses an adversarial loss function where the generator minimizes the discriminator's ability to distinguish between real and synthetic records, while the discriminator maximizes its accuracy. Over time, this iterative process enables the generator to learn the underlying data distribution and produce synthetic data that closely resembles real world patient records.

While GAN have demonstrated impressive capabilities, certain limitations have been identified. One notable challenge involves the generation of discrete data, which consists of distinct, countable values, as the original GAN framework is primarily designed to

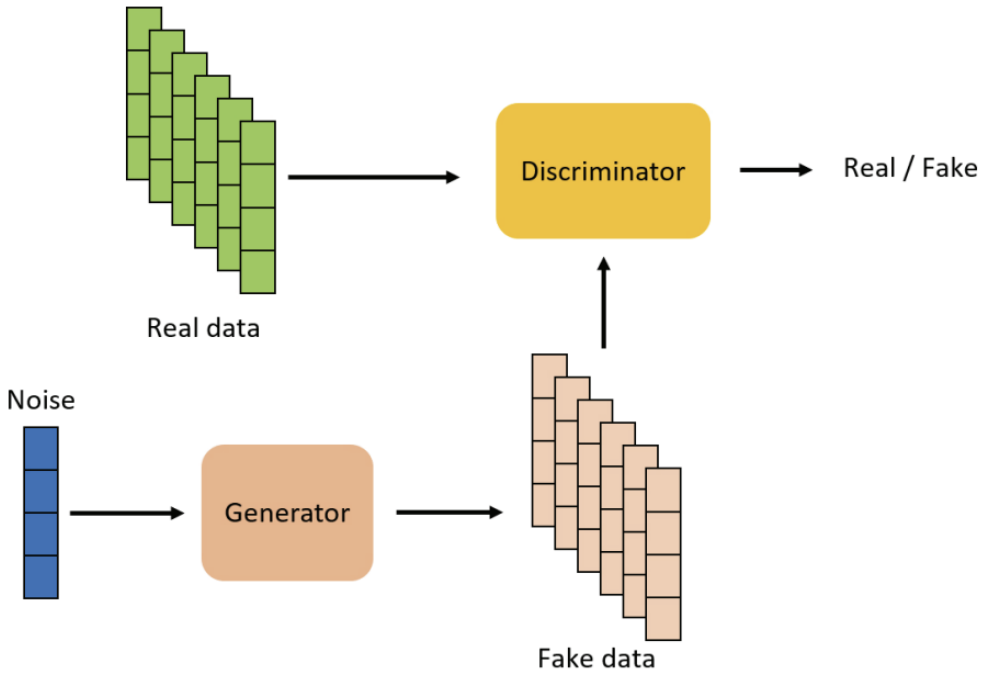


Figure 2.3: GAN model for synthetic data generation [36].

handle continuous data distributions. Additionally, concerns have been raised regarding privacy, as GAN may inadvertently expose sensitive information from the training data [10, 16]. This issue arises because GAN, especially when overfitted, can memorize specific patterns or instances from the original dataset, leading to the risk of data leakage. Under Membership Inference Attacks (MIA), GAN may leak information about the real data, as attackers might be able to identify that certain synthetic data points were generated based on specific real training data [16].

To address these challenges, several methodologies have been proposed to extend the original GAN framework, each targeting specific issues. These extensions will be discussed in the following topics, focusing on their unique contributions and use cases:

- **medGAN**: Designed for generating tabular healthcare data by modeling discrete and high dimensional variables.
- **CTGAN**: Improves GAN performance on tabular data by addressing imbalances and capturing complex feature relationships.
- **DP-ACTGAN**: Enhances privacy in tabular data generation by incorporating differential privacy mechanisms.
- **DoppelGANger**: Extends GAN applications to sequential and time series data by explicitly modeling temporal and contextual dependencies.

a) medGAN

medGAN (medical Generative Adversarial Network) addresses the incapability of GAN learning the distribution of discrete variables [10]. It adapts the Generative Adversarial Network framework to become a neural network model capable of generating discrete and high dimensional tabular data, specifically in the context of healthcare.

medGAN is designed to generate multi-label discrete variables, such as those found in EHRs, which represent events like the diagnosis of a certain disease or the administration of a specific medication [10].

To achieve this, *medGAN* introduces an additional autoencoder alongside the adversarial framework of GAN. This autoencoder is responsible for learning the underlying distribution of discrete features, such as diagnosis or medication codes, by encoding and reconstructing the data before passing it to the GAN. By mapping the original high dimensional data to a latent space representation, the autoencoder facilitates a more structured and compressed input for the GAN's generator, improving its ability to model sparse and complex datasets. This step supports the generator in learning complex dependencies among features, enabling the generation of realistic multi-label discrete samples [10].

medGAN also introduces an innovative technique called minibatch averaging to address a common challenge in GAN training - mode collapse. Minibatch averaging improves upon previous methods like minibatch discrimination, helping *medGAN* to produce more varied and representative synthetic data [10]. Additionally, *medGAN* has implications for data privacy, as it generates synthetic datasets that preserve the statistical properties of real world data without replicating individual patient records, reducing the risk of privacy violations [12].

However, *medGAN* remains vulnerable to mode collapse, particularly due to the challenges posed by its discrete and high-dimensional data structure. Since the adversarial training process relies on a feedback loop between the generator and the discriminator, the generator may struggle to capture the full diversity of categorical medical features, leading it to converge on a narrow subset of patterns rather than producing a broad range of synthetic samples. This issue is further exacerbated by the sparsity of medical data, where certain diagnostic codes and medication patterns appear more frequently than others, making it easier for the model to repeatedly generate a limited set of high probability samples rather than accurately modeling the entire distribution [10]. Another drawback is that medGAN faces difficulties when applied to complex datasets with high-dimensional features. As the complexity and dimensionality of the data increase, the model's ability to generate high-quality synthetic data declines [12].

b) CTGAN

CTGAN (Conditional Tabular GAN) is a GAN-based framework designed to address challenges associated with generating tabular data, particularly when dealing with imbalanced categorical variables and complex feature dependencies [13]. Unlike traditional GAN, CTGAN introduces the ability to handle the unique characteristics of tabular datasets.

CTGAN utilizes a conditional sampling mechanism to ensure balanced representation of categorical variables during training. This prevents the generator from overlooking underrepresented categories, a common issue in datasets with imbalanced distributions.

Additionally, CTGAN employs a mode-specific normalization process to capture complex relationships between features more effectively [13]. CTGAN introduces various new techniques [13]:

- **Conditional Generator**: Allows the generator to conditionally sample data based on specific feature values, ensuring diversity and balance across categories.
- **Mode-Specific Normalization**: Transforms data in a way that preserves interdependencies between features, improving the model's ability to learn complex patterns.
- **Training by Sampling**: During training, CTGAN samples data points by selecting a single column and conditioning the generator on the column's values. This process ensures that the model learns the conditional distributions of the features, thereby capturing both categorical and continuous variable dependencies effectively.

The CTGAN framework has proven effective for generating high-quality synthetic tabular data that retains the statistical properties of the original dataset while maintaining feature relationships. It is particularly useful in applications where privacy concerns limit access to sensitive data, such as in healthcare data [13].

Despite its advancements, CTGAN has certain limitations. One significant challenge lies in modeling extremely imbalanced data distributions. While the conditional sampling mechanism improves representation for underrepresented categories, CTGAN may still underperform in cases of imbalance or highly sparse data. Additionally, as with many GAN-based approaches, CTGAN requires heavy computational resources and can have extended training times due to its complex architecture and sampling methods [13].

While CTGAN is designed for static tabular data, frameworks like DP-ACTGAN and DoppelGANger extend these principles to include privacy preserving mechanisms and sequential data generation, respectively.

c) DP-ACTGAN

DP-ACTGAN (Differentially Private - Auxiliary Classifier Conditional Tabular GAN) was developed to address the privacy concerns associated with traditional GAN, particularly their vulnerability to leaking sensitive information from the training dataset. Traditional GAN are susceptible to Membership Inference Attacks, where attackers might identify that certain synthetic data points were generated based on specific real training data [16]. This vulnerability arises from the potential of GAN to overfit the training data, leading to the inadvertent memorization and disclosure of private details, which can compromise the privacy of individuals represented in the original data [16].

To mitigate this risk, DP-ACTGAN integrates differential privacy mechanisms by adding Gaussian noise to the gradients during backpropagation. This ensures that

individual records in the training dataset cannot be reconstructed or inferred from the generated synthetic data. DP-ACTGAN employs a carefully calibrated privacy budget, measured in terms of ε (epsilon) and δ (delta), where ε controls the maximum influence that a single record can have on the output and δ allows for a small probability that this guarantee may not strictly hold. Together, these parameters balance the trade-off between privacy protection and data utility. By controlling the level of noise added during training, the model prevents data leakage while maintaining the usability of the generated data, making it suitable for applications requiring strict privacy guarantees [16].

In addition to its privacy preserving enhancements, DP-ACTGAN incorporates an auxiliary classifier within the GAN framework. This classifier supervises the generator by providing additional feedback during the training process, enabling the generation of high-quality and realistic samples. By addressing the dual objectives of privacy preservation and data quality, DP-ACTGAN effectively balances these often competing requirements. The integration of the auxiliary classifier and differential privacy mechanisms, which are central to DP-ACTGAN's architecture, is illustrated in Figure 2.4.

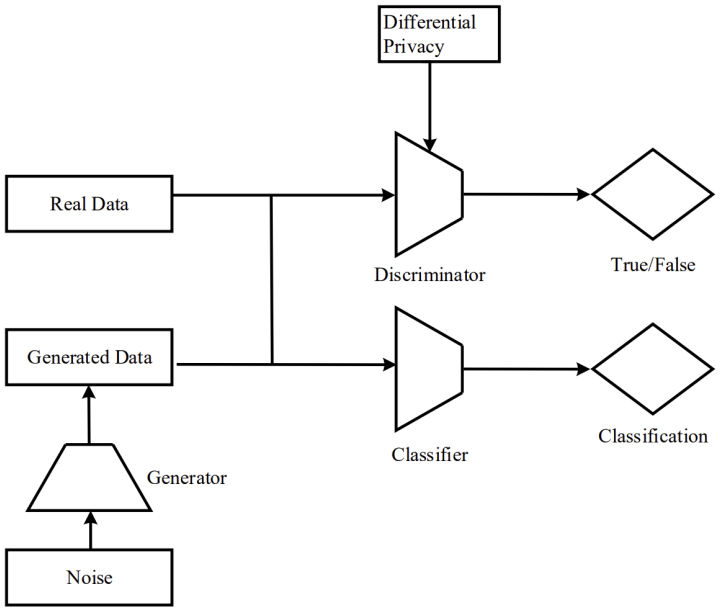


Figure 2.4: Structure of DP-ACTGAN [16].

Despite its advancements, DP-ACTGAN has certain limitations. The addition of differential privacy mechanisms introduces a trade-off between privacy and utility, where excessive noise may reduce the integrity of the generated data. This challenge is particularly evident in datasets with rare or underrepresented categories, as the model may struggle to accurately reflect these minority classes. Additionally, the computational complexity of DP-ACTGAN, due to its privacy mechanisms and auxiliary classifier, can result in longer training times and increased resource requirements, especially for large scale datasets. Lastly, DP-ACTGAN may face difficulties capturing long term dependencies in sequential or highly complex data, limiting its versatility for certain applications [16].

d) DoppelGANger

Traditional GAN often struggle with sequential data due to the intricate temporal dependencies and feature correlations within these datasets. Unlike medGAN and ACTGAN, which focus on static tabular data, for sequential and time-series data, *DoppelGANger* provides a solution. *DoppelGANger* builds upon the traditional GAN framework but explicitly separates static and dynamic components of data, enabling more robust modeling of time series. One of its key aspects is its ability to address challenges inherent to sequential data, such as temporal dependencies and feature correlations, while also mitigating mode collapse that occurs when the generator produces a limited variety of outputs, failing to represent the full diversity of the original data. Its main focus is on generating realistic, high-dimensional time series data, such as electronic health records, while addressing challenges like temporal dependencies, feature correlations and mode collapse [37].

DoppelGANger aims to retain the statistical properties of the original dataset while reducing the risk of direct replication, although the extent to which this is achieved remains limited. This approach reduces the risk of private information leakage, which, as observed in GAN, can occur if the model memorizes the training data [37]. DoppelGANger achieves this by ensuring diversity in the generated data, effectively avoiding mode collapse and by encouraging the generator to produce samples that are statistically similar, but not identical to the real data. The principal objective of the generator of DoppelGANger is to address the issue of capturing temporal correlations, correlations between features and attributes and aliviate mode collapse [37].

The generator of DoppelGANger revolves around a two part generator framework:

1. **Context Generator**: Generates static context features, such as metadata or attributes that remain constant throughout the sequence. These static features set the foundation and constraints for the time series data.

2. **Time-Series Generator**: Conditioned on the static context features generated earlier, this component produces sequential data points that are consistent with the provided context. This ensures that the temporal dynamics and dependencies are accurately modeled.

DoppelGANger employs a **discriminator** that evaluates both the realism of the generated time series and the coherence between static and dynamic features.

This adversarial training framework iteratively improves the generator's capacity to mimic real world data distributions, producing synthetic datasets that maintain temporal integrity and contextual accuracy [37].

The training process involves minimizing a loss function that captures both temporal integrity and statistical similarity between the generated data and the real data. By integrating context features into the generation process, DoppelGANger not only ensures realistic sequences, but also maintains interdependencies between variables, a critical requirement for high-dimensional time series datasets.

Despite its strengths, DoppelGANger faces challenges with extremely large scale or high-dimensional datasets due to the increased complexity of training GAN as data dimensionality grows [37].

2.2.3 Autoregressive Models

Autoregressive models are theoretical frameworks similar to Generative Adversarial Networks but focus on sequential dependencies, making them especially relevant for tasks like time series data generation. These models are a class of statistical and machine learning techniques designed to predict future values in a sequence based on its past values. They are particularly effective for sequential data, where temporal dependencies and correlations between data points are essential. The principal idea behind autoregressive models is that the current value of a sequence can be expressed as a linear combination of its previous values, often with an added noise term to account for variability [32, 38].

In the context of synthetic data generation, autoregressive models are well suited for time series data, such as patient records. They are particularly advantageous when data is limited, as they can be used to augment training datasets by generating additional sequences that capture the observed distributions. This capability is especially useful in healthcare, where datasets often suffer from sparsity and a lack of sufficient data for model training. These models also stand out at capturing the temporal dependencies inherent in sequential datasets, making them a notable tool for tasks requiring high fidelity and consistency in time-dependent data [11].

The general autoregressive process can be mathematically expressed as:

$$X_t = c + \sum_{i=1}^{p} \phi_i X_{t-i} + \epsilon_t$$

where:

- X_t is the current value;
- X_{t-i} represents past observations;
- c is a constant;
- p is the number of lagged terms (order of the model);
- ϕ_i are coefficients for the lagged terms;
- ϵ_t is a noise term, capturing unexplained variations in the series.

In healthcare, autoregressive models can generate sequential synthetic data by modeling the progression of diseases, treatment plans or patient health records over time. By leveraging these models, it can produce realistic and consistent datasets while preserving privacy. However, autoregressive models face challenges [39], such as:

- **Data Sparsity**: Difficulty handling sparse or irregularly sampled data.
- **Scalability**: Increased computational requirements for high-dimensional sequences.
- **Long-Term Dependencies**: Limited ability to capture long term relationships in sequences. While modern variants like RNNs and Transformers have significantly improved this capability, challenges such as vanishing gradients (in RNNs) and high computational costs (in Transformers) remain.

a) ParSynthesizer and the Synthetic Data Vault Framework

ParSynthesizer is an advanced probabilistic autoregressive model developed within the SDV framework. Designed for synthetic data generation, *ParSynthesizer* extends traditional autoregressive principles to support multi-sequential data, making it particularly suited for timeseries and sequential datasets such as patient health records [15].

The SDV framework provides a comprehensive toolkit for generating, evaluating and working with synthetic data. *ParSynthesizer* uses the probabilistic modeling capabilities of SDV to generate data that captures temporal dependencies while maintaining statistical fidelity to the original dataset. This makes it a good tool for applications in healthcare, where preserving the sequential nature of data is important for tasks like simulating disease progression or treatment outcomes.

ParSynthesizer supports multi-sequential data, enabling the modeling of complex temporal relationships across multiple sequences. It integrates privacy preserving mechanisms to ensure that the generated synthetic data does not reveal sensitive information from the original dataset. Additionally, it provides flexibility in generating high-dimensional time series data, such as electronic health record sequences and financial time series, making it a adaptable solution for various domains.

In addition to *ParSynthesizer*, the SDV framework includes models like *CPAR* (Conditional Probabilistic Autoregressive Model). CPAR builds upon the foundations of *ParSynthesizer* by introducing conditional dependencies between variables, further improving the realism and utility of the generated synthetic data. This is particularly beneficial in healthcare contexts, where variables like patient demographics, diagnoses and treatments are highly interdependent [15].

While *ParSynthesizer* has several strengths, it inherits several typical limitations of autoregressive models, such as scalability challenges with large or high-dimensional datasets, difficulties in handling sparse or irregularly sampled data and struggles to capture long-term dependencies when temporal gaps are significant.

2.2.4 Synthea and Related Tools

Synthea[1] is an open source software designed to generate synthetic patients and their associated electronic health records.

[1] https://syntheticahealth.github.io/synthea/

The approach adopted by Synthea involves modeling real-world clinical workflows by simulating the lifecycle of virtual patients, from birth to death, generating data that reflects typical interactions within the healthcare system. These interactions include the prescription of medications, disease progression and health outcomes [9].

Synthea relies on publicly available health statistics, clinical guidelines and demographic information to inform its models, ensuring that the generated data remains consistent with actual population-level health statistics [9].

This focus on population level realism makes Synthea a valuable asset for testing healthcare systems, conducting public health studies and supporting research while safeguarding privacy.

a) PaDARser: Enhancing Synthea with Privacy Preservation

A core component of Synthea is *PaDARser* (Publicly Available Data Approach to the Realistic Synthetic EHR), a modular tool for data parsing and transformation. PaDARser plays an important role in integrating structured data sources into Synthea's framework. Its primary focus is on privacy preservation, ensuring that sensitive information is never compromised. It does this by assuming that access to real EHRs is impossible, relying solely on publicly available datasets [9]. By transforming real world datasets into anonymized forms, PaDARser guarantees that the generated data is free from any linkage to identifiable individuals while maintaining statistical realism. This privacy-focused design enhances Synthea's ability to generate synthetic data suitable for research and innovation without risking patient confidentiality. However, as a supporting tool, PaDARser functions within the Synthea ecosystem rather than as a standalone synthetic data generator.

b) Validation and Limitations of Synthea

The validity of Synthea's synthetic data has been assessed in studies [40] comparing its output against real-world clinical data using clinical quality measures, such as those defined by the Centers for Medicare and Medicaid Services [40].

The findings showed that Synthea closely mirrors real world data in areas like chronic disease prevalence and treatment patterns. However, deviations were noted, including demographic over representation and limited variability in rare conditions. Additionally, Synthea does not account for EHRs deviations from post-service care, limiting its ability to model heterogeneous health outcomes [40].

While Synthea is effective for simulating population level trends and general patterns, these limitations suggest that caution should be taken when applying its data to research scenarios that require detailed modeling of individual level clinical variability.

2.2.5 Markov Chains

Markov Chains are mathematical models used to describe systems that evolve over time in a probabilistic manner, where the next state depends solely on the current state, a property known as the Markov assumption.

a) Hidden Markov Models

Hidden Markov Models (HMMs) extend basic Markov Chains by introducing hidden (latent) states that emit observable outputs according to probabilistic rules. In the context of synthetic data generation, they are particularly suitable for longitudinal data such as patient health records [33].

Each patient sequence is modeled as a series of transitions between hidden disease states, capturing the underlying latent progression of health conditions over time. At each time step, the observable data is generated from a probability distribution that is conditioned on the current hidden state, allowing the model to account for the stochastic nature of clinical observations.

Typical Workflow:

1. Preprocess data: normalize continuous features and encode categorical ones.
2. Fit the model: learn transition and emission probabilities from real sequences.
3. Generate new sequences: sample hidden state paths and emit observations.
4. Postprocess: revert normalizations and encodings.

b) Hybrid Markov-GAN Models: EHR-M-GAN

To overcome the limitations of classical Markov models in representing the complex and heterogeneous nature of EHRs, Jin et al. [41] introduced a hybrid generative framework known as *EHR-M-GAN*. This model combines Generative Adversarial Networks (GANs) with RNNs, where the RNNs component acts as a temporal proxy to capture Markovian dynamics implicitly. By doing so, the model is capable of generating realistic patient trajectories that reflect both the temporal structure and variability of real-world EHR data.

EHR-M-GAN is specifically designed for longitudinal datasets containing a mixture of categorical, continuous and temporal variables. Unlike traditional models that rely heavily on one-hot encoding, EHR-M-GAN directly models raw data formats, increasing efficiency and reducing dimensionality.

2.2.6 Multi Table Models

Recent advances in synthetic data generation have introduced models capable of preserving relationships across multiple tables. The `SDV` library [34] provides a suite of models for this purpose, including:

- **Day Z Synthesizer**: Generates data based solely on metadata. Ensures schema validity and referential integrity but does not learn statistical distributions.
- **Independent Synthesizer**: Trains each table independently using machine learning. Offers speed and scalability but disregards relational consistency.
- **HSA Synthesizer**: A segment-based model that ensures both speed and relational integrity, although available only in the enterprise edition.
- **HMA Synthesizer**: A hierarchical model that captures table relationships from real data. It is the most complete model available in the open-source version and supports up to 5 related tables with limited relational depth.

Given the objective of modeling clinical data involving several interrelated entities the `HMA Synthesizer` stands out for its ability to model hierarchical structures in a statistically meaningful way while remaining compatible with the open-source ecosystem.

2.3 Discussion

This section examines the main methodologies introduced in Section 2.2, focusing on their potential applicability to the generation of synthetic Electronic Health Records. Given the sequential nature of EHRs data, this analysis evaluates the most suitable methods by considering their ability to preserve temporal dependencies and address privacy concerns.

2.3.1 Overview of Approaches

The reviewed methodologies for synthetic data generation fall into two main categories:

- **Statistical and Simulation Based Approaches:**
  - **Synthea**: A healthcare focused simulation tool that generates synthetic EHRs based on population health trends and clinical workflows. While effective for modeling population level data, Synthea's dependence on publicly available statistics and predefined rules limits its ability to accurately capture unique patient level details and nuanced temporal patterns.
  - **PaDARser**: Enhances Synthea by anonymizing datasets and preserving statistical realism. However, its application is restricted to the Synthea ecosystem and it does not independently address sequential data challenges.

- **Machine Learning Based Approaches:**
  - **Generative Adversarial Networks (GANs)**: GAN based methods like medGAN, CTGAN, DP-ACTGAN and DoppelGANger offer advanced capabilities for generating synthetic data. While medGAN and CTGAN are suitable for tabular data, DoppelGANger explicitly models sequential and time-series data, making it a strong methodology for EHRs generation. ACTGAN, while primarily designed for tabular data with logical dependencies, can also be explored for EHRs generation with some necessary adjustments to accommodate sequential patterns or temporal dependencies.
  - **Autoregressive Models**: ParSynthesizer in the Synthetic Data Vault framework excels at modeling temporal dependencies, making them highly relevant for EHRs. Their probabilistic approach ensures the generated data reflects the sequential nature of real-world records.

2.3.2 Promising Methodologies

The main synthetic data generation methodologies discussed in this chapter are summarized in Table 2.1, which presents their key advantages and limitations. This comparison provides insights into the suitability of each approach for Electronic Health Record generation, emphasizing their ability to preserve temporal dependencies and ensure data privacy, both critical factors for producing realistic and reliable synthetic health records.

Table 2.1: Comparison of Synthetic Data Generation Methodologies

Methodology	Advantages	Limitations
Synthea	Tailored for healthcare workflows. Provides realistic population-level data.	Limited individual variability. Can not model complex temporal dependencies.
PaDARser	Enhances privacy within Synthea. Ensures compliance with anonymization standards.	Limited applicability beyond Synthea. Relies on publicly available datasets.
medGAN	Focused on high dimensional, discrete tabular data.	Prone to mode collapse. Ineffective for sequential data.
CTGAN	Handles imbalanced and complex tabular data effectively. Preserves feature relationships.	Limited to static data. Computationally intensive.
DP-ACTGAN	Integrates differential privacy for strong privacy guarantees. Generates high-quality tabular data.	Complexity increases with privacy constraints. Limited applicability to sequential data.
DoppelGANger	Explicitly models sequential data. Separates static and dynamic components.	Computationally demanding. Challenges with very high dimensional datasets.
ParSynthesizer	Probabilistic modeling of temporal data.	Lacks healthcare specific optimizations.

Among the listed methodologies, considering the sequential nature of EHRs data, **DoppelGANger**, **ParSynthesizer** and **ACTGAN** emerge as the most promising approaches:

- **DoppelGANger**: This model's ability to explicitly separate static and dynamic features and handle time series data aligns well with the structure of EHRs. Its robust approach to maintaining temporal dependencies ensures that synthetic EHRs remain realistic and usable for downstream applications.

- **ACTGAN**: Although ACTGAN is primarily tailored for generating tabular data with logical dependencies, it can be adapted for EHRs generation by incorporating mechanisms to capture temporal dependencies. Logical dependencies refer to relationships dictated by rules or constraints (e.g., a patient's diagnosis influencing prescribed medications), which ACTGAN is adept at preserving. However, EHRs data often involves temporal dependencies, such as the progression of a patient's vital signs or the timing of medical interventions. Adjustments to ACTGAN, such as integrating time series modeling techniques or conditioning on sequential features, could enable it to generate synthetic EHRs that respect both logical and temporal dependencies, making it a promising area for exploration.

- **ParSynthesizer**: With its probabilistic and autoregressive capabilities, ParSynthesizer excels in modeling sequential data. While it was not designed for the healthcare domain specifically, its flexibility allows it to adapt to various contexts, including EHRs generation.

2.4 Evaluation Metrics

The effectiveness of synthetic data generation techniques must be rigorously assessed to ensure the generated data retains statistical integrity, privacy protection and practical usability. To achieve this, multiple evaluation frameworks have been developed. This section outlines evaluation methodologies for assessing synthetic data, detailing their objectives, principal characteristics and the specific metrics used to evaluate integrity, privacy and utility.

The two main evaluation frameworks considered in this work are:

- **Gretel Library** [42]: A cloud-based synthetic data generation platform that includes built-in evaluation metrics for data quality, privacy and utility.

- **SDMetrics Library** [43]: A component of the Synthetic Data Vault ecosystem, designed to statistically validate synthetic data across multiple dimensions.

Both libraries offer complementary evaluation metrics, ensuring a comprehensive analysis of the generated synthetic datasets.

2.4.1 Integrity

Integrity refers to synthetic data's ability to maintain the statistical properties and logical consistency of the original dataset. To evaluate integrity, complementary evaluation metrics from the **Gretel Library** and the **SDMetrics Library** provide a structured assessment framework. A primary measure of statistical integrity is the Synthetic Quality Score (SQS) from Gretel, which is defined as [42]:

> "The quality score is an estimate of how well the generated synthetic data maintains the same statistical properties as the original dataset."

In this sense, the SQS can be interpreted as an integrity or confidence score, indicating whether scientific conclusions drawn from the synthetic dataset would likely align with those derived from the original data.

The SQS assesses the overall statistical integrity of synthetic data by analyzing field distributions, correlations and structural stability. Additionally, SDMetrics' Correlation-Similarity provides a focused evaluation of how well the relationships between features are preserved, ensuring that core dependencies remain intact. The following metrics outline the principal aspects of integrity and their individual components:

- **Synthetic Quality Score [Gretel]**: A composite score that evaluates how well synthetic data preserves the statistical properties of the original dataset. It is calculated as a weighted combination of three key metrics:
  - **Field Correlation Stability**: Measures how well correlations between feature pairs in the original dataset are preserved in synthetic data. This is evaluated by calculating correlation differences between the real and synthetic datasets, with lower differences indicating higher Field Correlation Stability.
    Ideal score: 1.0 - all pairwise feature correlations are preserved exactly.
  - **Deep Structure Stability**: Assesses how well synthetic data preserves complex feature relationships using Principal Component Analysis (PCA) [42]. PCA reduces the dataset to a few principal components, capturing its essential structure. Gretel compares the distributional distances of these components between real and synthetic datasets, with smaller distances indicating higher Deep Structure Stability. This metric provides quick feedback on the suitability of synthetic data for machine learning applications.
    Ideal score: 1.0 - PCA component distributions match between real and synthetic data.
  - **Field Distribution Stability**: Evaluates how closely the individual field distributions in the synthetic data match those in the original dataset. This is measured using the Jensen-Shannon Distance (JS Distance), a common metric for comparing probability distributions. The score is defined as $1 - JSDistance$, meaning that values closer to 1.0 indicate stronger alignment between the real

and synthetic distributions.
Ideal score: 1.0 - field distributions are identical (JS Distance = 0).

- **CorrelationSimilarity [SDMetrics]**: Evaluates the extent to which the synthetic dataset maintains the correlation structure of the original data, ensuring that relationships between features remain consistent.
  Ideal score: 1.0 - indicating that the correlations are preserved exactly.

2.4.2 Privacy

Ensuring privacy in synthetic data is critical to prevent adversarial attacks and unauthorized re-identification of individuals. Privacy risks can be assessed using metrics from both the **Gretel Library** and the **SDMetrics Library**.

The Data Privacy Score from Gretel serves as a primary measure, evaluating the extent to which synthetic data prevents privacy leaks. This score is computed by analyzing two types of adversarial attacks: Membership Inference and Attribute Inference. Additionally, SDMetrics' privacy risk measures, such as Categorical CAP and DisclosureProtection, assess the potential for sensitive information disclosure in synthetic data. Privacy protection will be assessed using the following metrics:

- **Membership Inference Protection [Gretel]**: Evaluates how well the synthetic data defends against membership inference attacks. These attacks occur when an adversary attempts to determine whether specific data points were part of the model's training set by exploiting differences in the model's responses to training versus unseen data points. A high Membership Inference Protection score indicates that the synthetic data effectively protects the original data against such attacks.
  Ideal score: 1.0 - model reveals no membership information.

- **Attribute Inference Protection [Gretel]**: Assesses the resistance of synthetic data to attribute inference attacks. In these attacks, adversaries attempt to infer sensitive or missing attributes about individuals in the dataset using known attributes and the model's output. A high Attribute Inference Protection score reflects strong protection against such attacks.
  Ideal score: 1.0 - no sensitive attributes can be inferred from synthetic data.

- **Personally Identifiable Information Replay [Gretel]**: Evaluates how often personally identifiable information from the original dataset appears in the synthetic data. A lower replay count indicates better privacy protection, with rarer entities (e.g., full names, addresses) expected to have minimal reproduction.
  Ideal score: 0.0 - no personally identifiable information is reproduced.
  Note: Unlike the other privacy metrics where higher is better, for PII Replay lower values indicate stronger privacy.

- **Categorical Attribute Privacy (CAP) [SDMetrics]**: This metric assesses the privacy of categorical attributes in synthetic datasets by evaluating the likelihood of re-identifying individuals based on these attributes. The Categorical CAP measures the risk of disclosing sensitive information through an inference attack. It evaluates how well synthetic data conceals sensitive categorical variables while preserving utility. Higher scores indicate stronger privacy for categorical attributes.
  Ideal score: 1.0 - perfect privacy protection for categorical attributes.

- **DisclosureProtection [SDMetrics]**: This metric simulates scenarios where an attacker uses known attributes to infer unknown sensitive information. Higher scores indicate stronger privacy protection. Based on the Categorical Attribute Privacy (CAP) framework, it assesses disclosure risks, with an estimated version (DisclosureProtectionEstimate) used for large datasets.
  Ideal score: 1.0 - maximum protection against attribute disclosure attacks.

2.4.3 Utility

Assessing the utility of synthetic data is essential to ensure its applicability in machine learning and statistical analysis. The **Gretel Library** and **SDMetrics Library** provide metrics to assess how well synthetic data replicates real data while remaining viable for predictive modeling. The Machine Learning Quality Score (MLQS) from the Gretel library compares ML performance on synthetic vs. real data, while SDMetrics' Statistic Similarity and Boundary Adherence assess alignment in descriptive statistics and value distributions. The following metrics structure this evaluation:

- **Machine Learning Quality Score [Gretel]**: Measures how well synthetic data is suitable for ML applications by comparing model performance on real and synthetic datasets. The process involves training multiple models, aggregating top-performing results and computing MLQS as the ratio of synthetic-to-real model performance. A higher MLQS score indicates strong predictive power, ensuring the synthetic data's usability for Machine Learning training, experimentation and deployment.
  Ideal score: 1.0 - predictive performance on synthetic data matches that of real data.

- **Statistic Similarity [SDMetrics]**: This metric compares summary statistics such as mean, median and standard deviation, between real and synthetic data columns, ensuring the synthetic data preserves the original dataset's key statistical properties.
  Ideal score: 1.0 - summary statistics (mean, median, std) are identical.

- **Boundary Adherence [SDMetrics]**: This metric verifies whether the synthetic data adheres to the minimum and maximum boundaries of the original dataset. It ensures that the generated values fall within valid ranges, preserving data integrity.
  Ideal score: 1.0 - all synthetic values lie within valid boundaries.

2.4.4 Summary of Evaluation Approach

In this thesis, evaluation metrics from both Gretel and SDMetrics are applied to all three methodologies (*ACTGAN, DoppelGANger and ParSynthesizer*) to ensure a robust comparison.

- **Integrity**: Assessed through SQS, correlation stability, PCA and JS Distance.
- **Privacy**: Evaluated via membership inference tests, attribute inference protection and disclosure risks.
- **Utility**: Measured using machine learning performance metrics and statistical similarity.

This structured evaluation allows for a rigorous and comparative assessment of synthetic data generation methodologies, enabling a well-founded determination of their suitability for generating synthetic versions of the High Users dataset while balancing integrity, privacy and utility.

3

Data Analysis

This section presents the dataset associated with the thesis, which was analyzed and explored to comprehend its structure and characteristics, which later facilitated the identification of the most suitable methodologies for the defined objectives. Building on this knowledge, a more extensive data analysis was conducted and an Entity–Relationship (ER) model was developed. The primary objective of this model is to simplify and structure the data for the next step, which is the practical implementation.

3.1 Dataset Description

The data used and analyzed in this study was collected by *VOH.CoLAB* [44] in collaboration with "Hospital Garcia da Orta," focusing specifically on the Emergency Department (Urgências). The dataset has patients' records classified as High Users (HU), defined as individuals who visited the Emergency Department more than ten times per year during the period from June 2016 to February 2020. This data includes demographic details and hospital service utilization information for a total of 972 patients [44].

The data is divided among five different CSV files:

- **Consulta_Episodios_analises_exames_diagnosticos_medicamento**: This file contains detailed records of medical appointments attended by patients at "Hospital Garcia da Orta". It includes information about the date and type of medical appointments, the medical specialities involved, the analyses and exams performed, as well as diagnoses and prescribed medications associated with the medical appointments. Additionally, it provides details such as prescription dates and dosages.

- **Doentes_obito**: This file includes a diverse range of data about the patients. It contains their personal, demographic and clinical information. It also contains data about the patients who passed away during the data collection period.

- **Internamento_Episodios_analises_exames_gdh**: This file provides information about patients who were hospitalized at "Hospital Garcia da Orta", including the

admission date, discharge date, the number of analyses and exams performed during their stay, their diagnoses and other relevant information.

- **Internamento_Medicamentos**: This file contains records of medications prescribed to patients during their hospital stay, including the prescription date, medication name, dosage, administration frequency, start and end dates and more information.

- **Urgencias_Episodios_Triagem_analises_exames_medicamento**: This file contains detailed records of emergency room visits. It includes information about patient identification, analyses and exams performed and medications prescribed during the visit. It specifically focuses on emergency episodes, capturing additional context such as the admission date, time of admission and triage details.

Additional exploratory data analysis was conducted to better understand the dataset and identify potential inconsistencies. This analysis includes a comprehensive assessment of missing values and other relevant statistics, which are detailed in Annex I - Data Analysis. Given the considerable scale of the dataset, both in terms of the number of records and attributes, significant effort was dedicated to thoroughly exploring its structure, distributions and anomalies in order to support informed modeling decisions.

3.2 The Construction of the Entity-Relationship Model

Based on the CSV files, an ER Model was constructed to demonstrate the relationships between the entities, as shown in Figure 3.1.

From the data, the following entities were created: `Patient`, `Medical Appointment`, `Hospitalization`, `Emergency`, `Exam`, `Laboratory Test`, `Diagnose`, `Medication` and `Prescription`.

To make the model easier to understand, the entity `Clinical Episode` was also created. A `Clinical Episode` can represent a `Medical Appointment`, a `Hospitalization`, or an `Emergency`. The entity `Clinical Episode` uses a disjoint `ISA` relationship to ensure that each clinical episode belongs exclusively to one of these categories.

The relationships between these entities can be summarized as follows: A `Patient` can have one or more `Clinical Episodes` and each `Clinical Episode` must be associated with a `Patient`. For `Medical Appointment`, `Hospitalization` and `Emergency`:

- They can each have zero or more `Exam` and `Laboratory Test` associated with them.

- A `Medical Appointment` or `Hospitalization` may have zero or one `Diagnose`, but a `Diagnose` may be linked to multiple `Medical Appointments` or `Hospitalization`.

- `Medication` can be prescribed for `Medical Appointment` and `Hospitalization`, with a relationship allowing zero or more `Medications` linked to them.

- `Prescription` can be issued during an `Emergency`, with zero or more `Prescriptions` linked to it.

`Exam` and `Laboratory Test` must always be associated with one `Clinical Episode` and cannot exist independently in the system. Similarly, `Prescription` and `Medication` can not exist in the system without being linked to one of the previously mentioned entities.

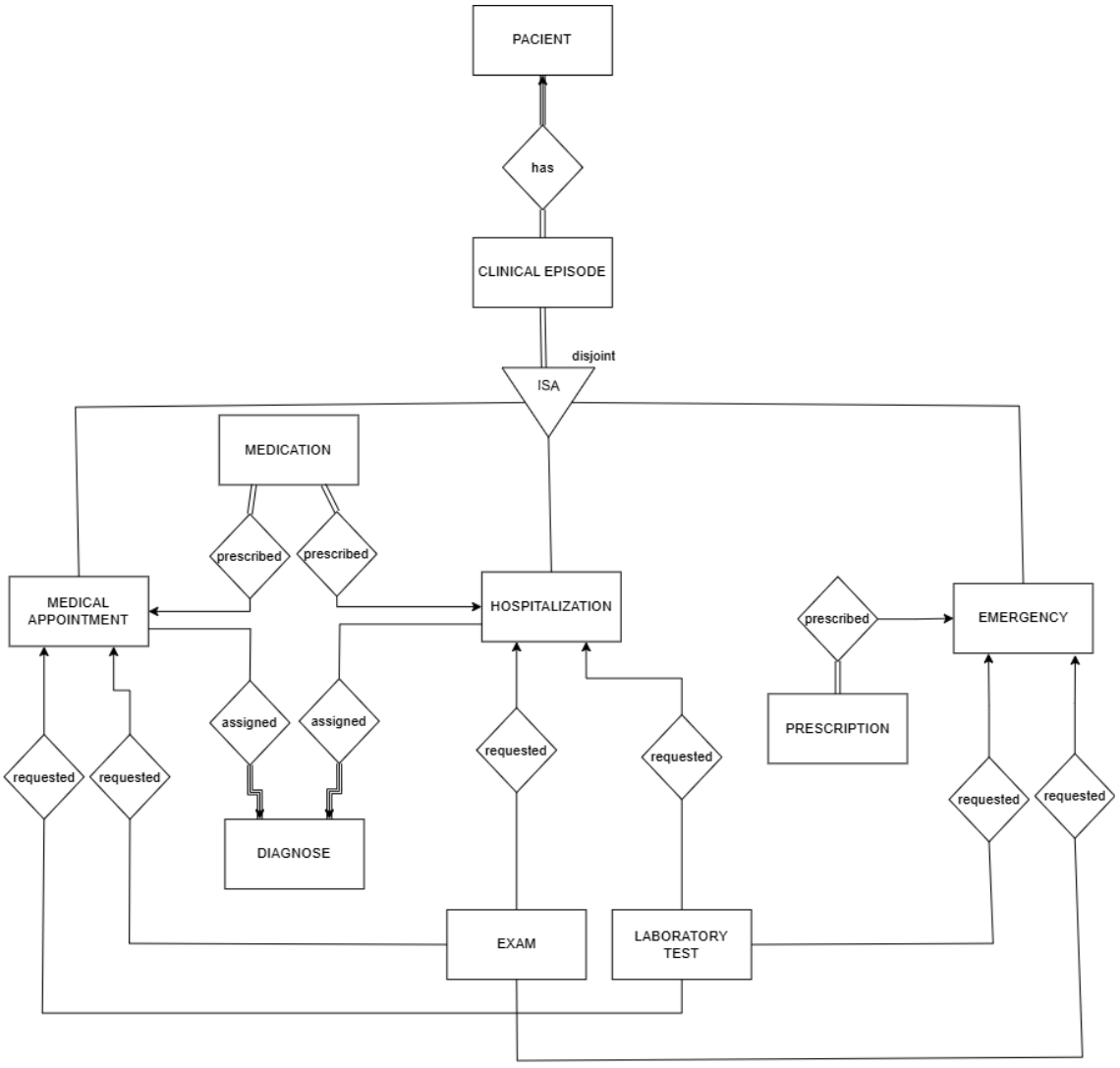


Figure 3.1: Entity-Relationship Model.

3.2.1 Important Considerations

To keep the model simple and avoid unnecessary complexity, the following entities from the CSV files were merged:

- There were initially two separate entities for `Medication`: one associated with `Medical Appointment` and the other with `Hospitalization`. These entities shared

all attributes except one - `OBS_MED`, which was rarely filled (present in only approximately 1.23% of the rows). Due to this significant overlap of attributes, the entities were merged into a single `Medication` entity to simplify the data structure and avoid redundancy.

- Similarly, there were two separate entities for `Diagnose`: one linked to `Medical Appointment` and another to `Hospitalization`. Due to the similarity in attributes, these were also merged into a single `Diagnose` entity.

Annex II - Mapping of Entities and CVS Files, presents the origin of each entity and provides a detailed mapping between their English and Portuguese names, as defined in the corresponding CSV files.

3.3 Data Constraints

To ensure data integrity and consistency in the database, the following constraints must be implemented. These represent only a subset of the rules that must be upheld, as the synthetic data generated by the selected methodologies must comply with these and other constraints to be considered valid and useful:

- **Deceased Patients Remain Inactive**: Patients cannot attend appointments after death, meaning that a patient recorded in the `Doentes_obito` dataset (marked as deceased) cannot have entries for medical appointments (`Consultas`) nor emergency episodes (`Urgência`) nor hospitalizations episodes (`Internamentos`) dated after their recorded date of death.
- **Hospitalization Dates**: The admission date in `Internamento` must be earlier than the discharge date.
- **Analyses, Exams and Medications**: No analyses, exams or medications can be associated with a hospitalization episode that has no recorded admission date.

3.4 Data Cleaning Process

Following the detailed description of the dataset, we proceeded with an analysis of the dataset's attributes to identify those that are most likely to be excluded based on their meaning and relevance.

As illustrated in Figure 3.1, which captures the relationships between entities and enables their grouping based on shared characteristics, the dataset under analysis is entirely tabular in nature. A considerable number of its attributes represent temporal information, which underscores the need for a synthetic data generation approach that can effectively model both structured tabular data and sequential temporal patterns. Nevertheless, the dataset also contains attributes that are incomplete, irrelevant or inconsistently filled. If

left unaddressed, such issues may compromise the integrity and practical utility of the synthetic data to be generated.

To address this, we examined each file in the dataset and identified several attributes that were either insufficiently filled, poorly structured or potentially detrimental to the synthesis process. To ensure the quality and reliability of the resulting synthetic dataset, such elements were reviewed and removed when deemed inadequate or potentially harmful to the generation process.

Below, we present a file by file summary of the attributes reviewed, detailing the reasoning behind each removal or retention decision.

3.4.1 `Doentes_Obito.csv`

- **HABILITACOES**: This attribute has only 22 filled values out of a total of 972 records, corresponding to just 2.26% of the data being populated. The column was removed because, with such a low fill rate, the model would either replicate this sparsity, resulting in an uninformative attribute or attempt to overfit noise during imputation.

- **FREGUESIA**, **CENTRO_SAUDE**, **MEDICO_FAMILIA**, **ISENCAO**: These attributes contain PII, which could be used to reidentify individual patients. While all attributes in the dataset involve some level of sensitive information, these particular fields pose a heightened privacy risk. Given that they were not essential for the case study, these attributes were removed to reduce the likelihood of reidentification.

- **Patients without associated Clinical Episodes**: A total of 52 patients had no associated records in consultations, hospitalizations, or emergency episodes and were therefore removed from the dataset.

3.4.2 `Internamentos_Medicamentos.csv`

- **SEQ_PH_MED_PRES**: Several duplicate entries were identified in this file, differing only in the `SEQ_PH_MED_PRES` attribute. This likely represents multiple prescriptions of the same medication within a single episode. While introducing a new column indicating the number of units prescribed (e.g., boxes) could reduce the duplication, only 271 duplicates were found out of a total of 213,031 records. That said, no correction was applied, since the impact was deemed negligible.

PRESCRICAO	SEQ_PH_MED_PRES	DATA_PRESC	INT_EPISODIO	MEDICAMENTO	NOME_MED	TIPO_MD	DOSE	UNID_MED	VIA_ADM	FREQ_PRES
6.769.643	16.860.290	20.03.31	9.442.330.187	F1390	Furosemida 20 mg/2 ml Sol in	Sol. inj.	100	MG	I.V.	Continua
6.773.272	16.868.855	20.04.02	9.442.330.187	F1390	Furosemida 20 mg/2 ml Sol in	Sol. inj.	100	MG	I.V.	Continua
6.773.272	16.868.856	20.04.02	9.442.330.187	ZZZ99	MEDICAMENTO DE AMBULATÓ	Comp.	1	UND	Oral	Pequeno Almoço
6.773.272	16.868.857	20.04.02	9.442.330.187	ZZZ99	MEDICAMENTO DE AMBULATÓ	Comp.	1	UND	Oral	Pequeno Almoço
6.773.272	16.868.858	20.04.02	9.442.330.187	ZZZ99	MEDICAMENTO DE AMBULATÓ	Comp.	1	UND	Oral	Pequeno Almoço

Figure 3.2: Duplicate entries in the `Internamentos_Medicamentos.csv` file.

- **Delimiter Errors**: A number of rows contained formatting issues caused by incorrect use of semicolons (e.g., inserting “0;5” instead of “0.5”), which shifted values into

adjacent columns. For instance, the value "5" was sometimes incorrectly interpreted as the medication unit rather than part of the dosage. Affected rows were identified and corrected by shifting the values back to their appropriate columns.

- **Inconsistency with Hospitalization Episodes**: By cross-referencing this file with the `Internamento_Episodios_analises_exames_gdh` file, we observed that many records referenced hospitalization episodes that did not exist. These entries were removed, reducing the number of records from 213,031 to 66,448.

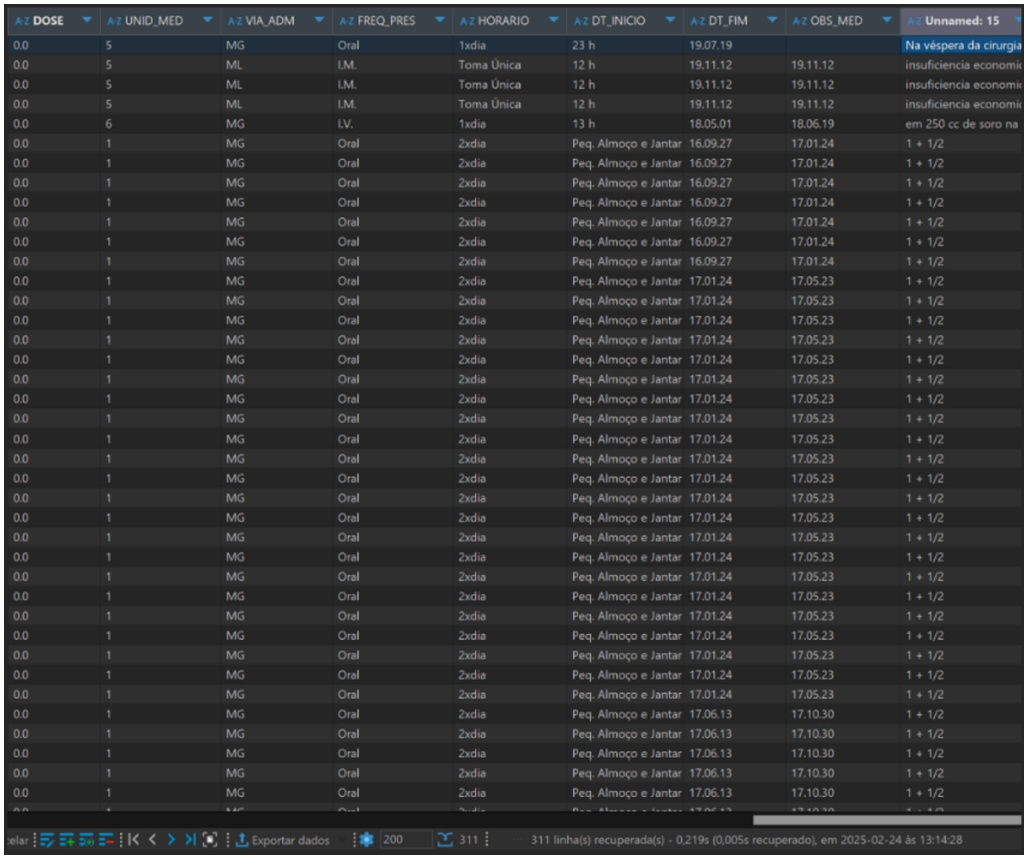
3.4.3 `Consulta_analises_exames_diagnosticos_medicamento.csv`

- **TIPO_DIAGNOSTICO**: Although only a small subset of consultation records included diagnostic information, all 486 diagnosis entries recorded in this dataset provided three elements: an identifier (`COD_DIAGNOSTICO`), a textual description (`DES_DIAGNOSTICO`) and a classification type (`TIPO_DIAGNOSTICO`). As shown in Table 3.1, while the other two attributes exhibited meaningful variation, `TIPO_DIAGNOSTICO` always held the same value: "Diagnóstico Principal" (the primary or main diagnosis assigned to the patient). Since this attribute does not add discriminatory value — being constant across all records — it was deemed redundant and removed to simplify the dataset.

Table 3.1: Summary statistics for the diagnostic attributes in consultation records.

Attribute	Count	Unique	Most Frequent (Top) / Frequency
`COD_DIAGNOSTICO`	486	71	274 / 301
`DES_DIAGNOSTICO`	486	71	GOTA / 301
`TIPO_DIAGNOSTICO`	486	1	Diagnóstico Principal / 486

- **DOSE misalignment and `Unnamed:15`**: A data entry error was detected where certain dosages were entered incorrectly, "0;5" instead of "0.5". This caused the "5" to shift into the next column, displacing all subsequent values. The column `Unnamed:15` appeared to have been introduced to capture overflow values in such cases, especially when the final field `OBS_MED` was pushed to the right. Further analysis revealed that the issue was not limited to "0.5mg" but extended to all sub-milligram values and other outliers (such as 0.1, 0.25, 0.5, 0.6 and 256.3). The affected rows were corrected by shifting values back into their correct columns and the `Unnamed:15` column was removed as it no longer served any purpose.



DOSE	UNID_MED	VIA_ADM	FREQ_PRES	HORARIO	DT_INICIO	DT_FIM	OBS_MED	Unnamed: 15
0.0	5	MG	Oral	1xdia	23 h	19.07.19		Na véspera da cirurgia
0.0	5	ML	I.M.	Toma Única	12 h	19.11.12	19.11.12	insuficiencia economi
0.0	5	ML	I.M.	Toma Única	12 h	19.11.12	19.11.12	insuficiencia economi
0.0	5	ML	I.M.	Toma Única	12 h	19.11.12	19.11.12	insuficiencia economi
0.0	6	MG	I.V.	1xdia	13 h	18.05.01	18.06.19	em 250 cc de soro na
0.0	1	MG	Oral	2xdia	Peq. Almoço e Jantar	16.09.27	17.01.24	1 + 1/2
0.0	1	MG	Oral	2xdia	Peq. Almoço e Jantar	16.09.27	17.01.24	1 + 1/2
0.0	1	MG	Oral	2xdia	Peq. Almoço e Jantar	16.09.27	17.01.24	1 + 1/2
0.0	1	MG	Oral	2xdia	Peq. Almoço e Jantar	16.09.27	17.01.24	1 + 1/2
0.0	1	MG	Oral	2xdia	Peq. Almoço e Jantar	16.09.27	17.01.24	1 + 1/2
0.0	1	MG	Oral	2xdia	Peq. Almoço e Jantar	16.09.27	17.01.24	1 + 1/2
0.0	1	MG	Oral	2xdia	Peq. Almoço e Jantar	16.09.27	17.01.24	1 + 1/2
0.0	1	MG	Oral	2xdia	Peq. Almoço e Jantar	17.01.24	17.05.23	1 + 1/2
0.0	1	MG	Oral	2xdia	Peq. Almoço e Jantar	17.01.24	17.05.23	1 + 1/2
0.0	1	MG	Oral	2xdia	Peq. Almoço e Jantar	17.01.24	17.05.23	1 + 1/2
0.0	1	MG	Oral	2xdia	Peq. Almoço e Jantar	17.01.24	17.05.23	1 + 1/2
0.0	1	MG	Oral	2xdia	Peq. Almoço e Jantar	17.01.24	17.05.23	1 + 1/2
0.0	1	MG	Oral	2xdia	Peq. Almoço e Jantar	17.01.24	17.05.23	1 + 1/2
0.0	1	MG	Oral	2xdia	Peq. Almoço e Jantar	17.01.24	17.05.23	1 + 1/2
0.0	1	MG	Oral	2xdia	Peq. Almoço e Jantar	17.01.24	17.05.23	1 + 1/2
0.0	1	MG	Oral	2xdia	Peq. Almoço e Jantar	17.01.24	17.05.23	1 + 1/2
0.0	1	MG	Oral	2xdia	Peq. Almoço e Jantar	17.01.24	17.05.23	1 + 1/2
0.0	1	MG	Oral	2xdia	Peq. Almoço e Jantar	17.01.24	17.05.23	1 + 1/2
0.0	1	MG	Oral	2xdia	Peq. Almoço e Jantar	17.01.24	17.05.23	1 + 1/2
0.0	1	MG	Oral	2xdia	Peq. Almoço e Jantar	17.01.24	17.05.23	1 + 1/2
0.0	1	MG	Oral	2xdia	Peq. Almoço e Jantar	17.01.24	17.05.23	1 + 1/2
0.0	1	MG	Oral	2xdia	Peq. Almoço e Jantar	17.01.24	17.05.23	1 + 1/2
0.0	1	MG	Oral	2xdia	Peq. Almoço e Jantar	17.01.24	17.05.23	1 + 1/2
0.0	1	MG	Oral	2xdia	Peq. Almoço e Jantar	17.01.24	17.05.23	1 + 1/2
0.0	1	MG	Oral	2xdia	Peq. Almoço e Jantar	17.01.24	17.05.23	1 + 1/2
0.0	1	MG	Oral	2xdia	Peq. Almoço e Jantar	17.01.24	17.05.23	1 + 1/2
0.0	1	MG	Oral	2xdia	Peq. Almoço e Jantar	17.01.24	17.05.23	1 + 1/2
0.0	1	MG	Oral	2xdia	Peq. Almoço e Jantar	17.01.24	17.05.23	1 + 1/2
0.0	1	MG	Oral	2xdia	Peq. Almoço e Jantar	17.01.24	17.05.23	1 + 1/2
0.0	1	MG	Oral	2xdia	Peq. Almoço e Jantar	17.01.24	17.05.23	1 + 1/2
0.0	1	MG	Oral	2xdia	Peq. Almoço e Jantar	17.01.24	17.05.23	1 + 1/2
0.0	1	MG	Oral	2xdia	Peq. Almoço e Jantar	17.01.24	17.05.23	1 + 1/2
0.0	1	MG	Oral	2xdia	Peq. Almoço e Jantar	17.01.24	17.05.23	1 + 1/2
0.0	1	MG	Oral	2xdia	Peq. Almoço e Jantar	17.01.24	17.05.23	1 + 1/2
0.0	1	MG	Oral	2xdia	Peq. Almoço e Jantar	17.01.24	17.05.23	1 + 1/2
0.0	1	MG	Oral	2xdia	Peq. Almoço e Jantar	17.06.13	17.10.30	1 + 1/2
0.0	1	MG	Oral	2xdia	Peq. Almoço e Jantar	17.06.13	17.10.30	1 + 1/2
0.0	1	MG	Oral	2xdia	Peq. Almoço e Jantar	17.06.13	17.10.30	1 + 1/2
0.0	1	MG	Oral	2xdia	Peq. Almoço e Jantar	17.06.13	17.10.30	1 + 1/2
0.0	1	MG	Oral	2xdia	Peq. Almoço e Jantar	17.06.13	17.10.30	1 + 1/2

Exportar dados 200 311 311 linha(s) recuperada(s) - 0,219s (0,005s recuperado), em 2025-02-24 às 13:14:28

Figure 3.3: Example of misalignment caused by using ";" instead of "," when entering dosage values.

- **TIP_CONSULTA**: This attribute contains only two values:"P" and "S" whose meanings are unknown. The absence of documentation or interpretation made it unreliable. Consequently, the column was removed.

3.4.4 `Urgencias_Triagem_analises_exames_medicamento.csv`

- **Unnamed: 5**: This attribute appeared to duplicate information already present in preceding columns. As it did not provide any additional value and contributed to redundancy, it was removed to simplify the structure of the dataset.

3.5 Data Preparation Process - Metadata

After completing the attribute-level cleaning of each file, based on relevance and the removal of redundant or non-informative fields, we proceeded to construct the Entity-Relationship (ER) model for the complete dataset. This model provides a clear overview of the remaining attributes and their corresponding metadata.

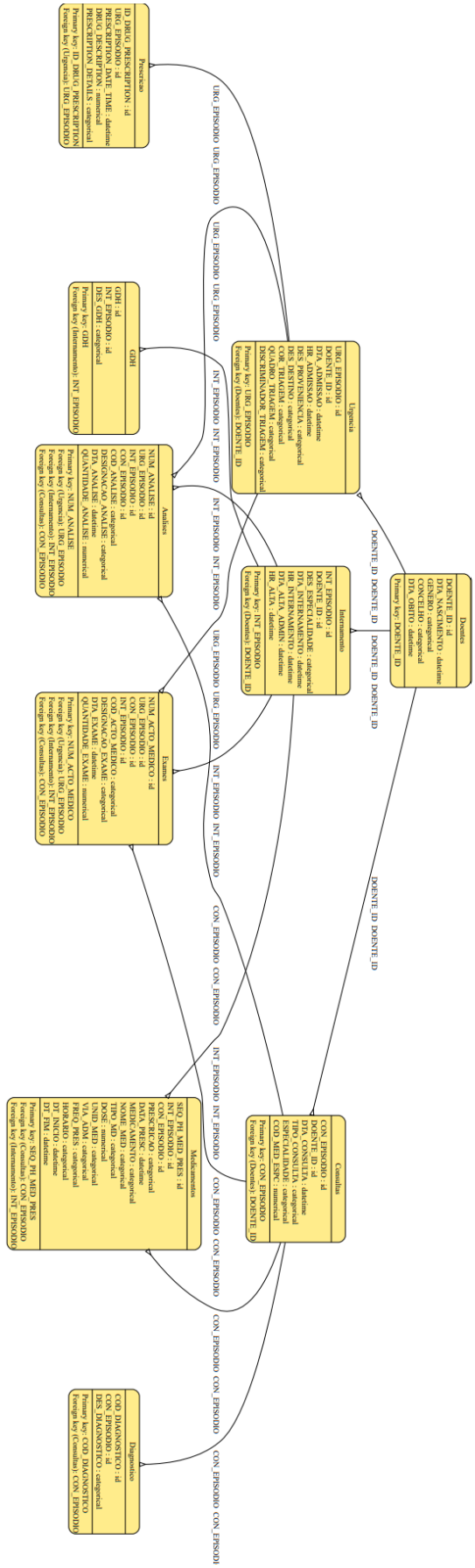


Figure 3.4: Final Metadata Diagram.

The metadata was first extracted, as it plays a crucial role in the structuring of the synthetic data. It ensures that relationships between entities and attributes are preserved and that the data generation process maintains consistency and coherence with the original dataset.

A preliminary version of the metadata diagram was automatically generated using the SDV API[1]. This initial output was subsequently revised manually to correct certain inconsistencies, most notably, adjustments to the data types of specific variables, resulting in the final version shown in Figure 3.4. The revision process also involved modifying attribute-level metadata, defining the relationships between tables and restructuring the dataset through the subdivision of certain tables where appropriate. Once these modifications were completed, the final metadata schema was validated using SDV's `validate` method to ensure consistency, integrity and compatibility with the underlying data.

Such revisions were expected, as the official SDV documentation itself warns that "detected metadata is not guaranteed to be accurate or complete and should be carefully reviewed and updated as necessary".

After obtaining the metadata, the ER model was constructed to visualize the dataset in a more organized manner, allowing for a clearer view of the attributes that remained after the data cleaning process, along with their corresponding metadata. Additionally, certain tables were simplified and merged in order to avoid redundancy and improve the overall structure of the dataset.

[1] https://colab.research.google.com/drive/1DlUlHNohxBhxN6GA5C_C4Fd9wx5AR11Y?usp=sharing

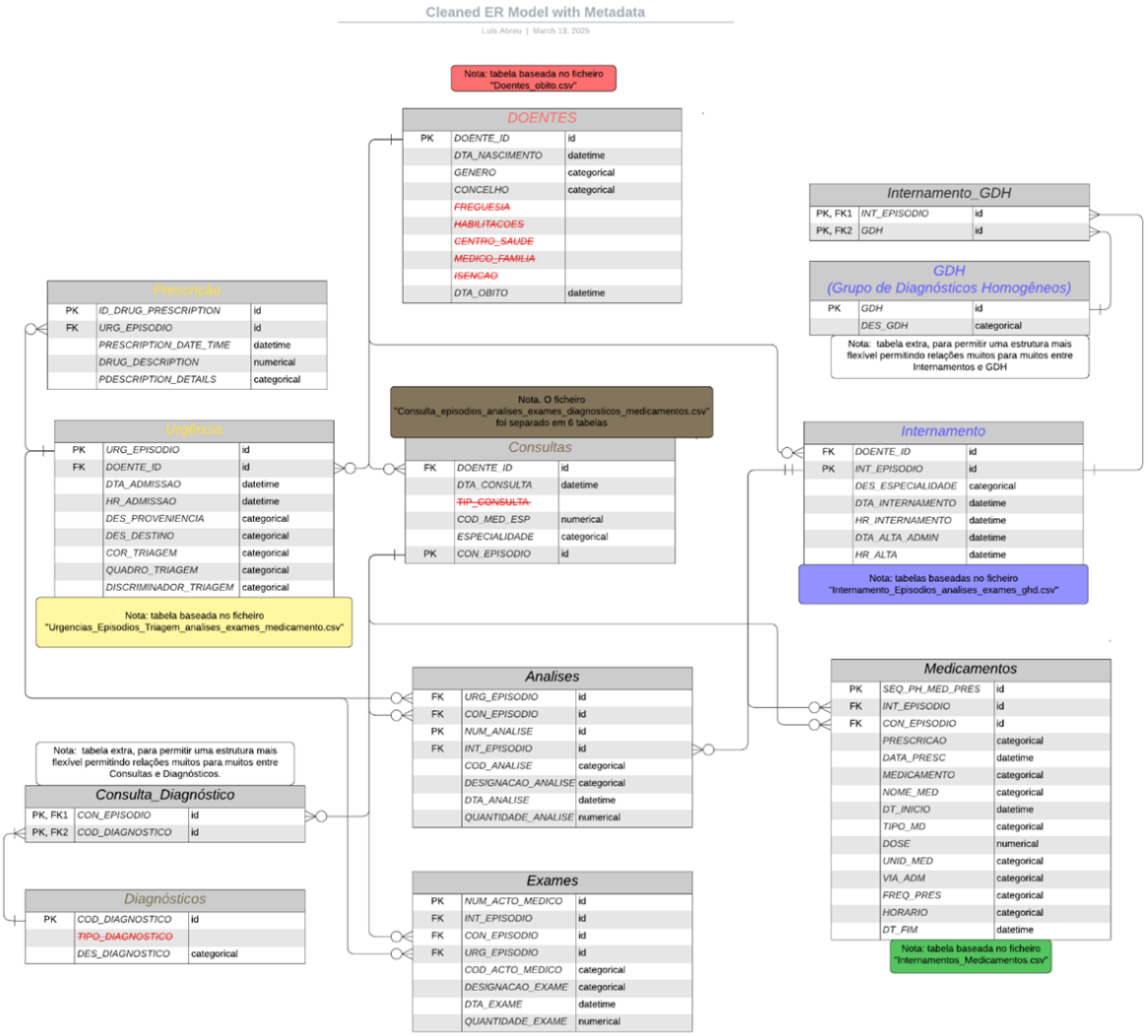


Figure 3.5: ER Model after cleaning and Metadata.

Based on the exploratory work conducted throughout this chapter, it becomes clear that the dataset under analysis presents specific characteristics that must be taken into account when choosing an appropriate synthetic data generation methodology.

Firstly, the input data, along with the synthetic data to be generated, consists of **tabular temporal sequences**, where each sequence corresponds to the clinical history of a single patient over a defined period. These sequences are composed of **multiple event types**, including medical appointments, hospitalizations and emergency episodes, each with their own associated diagnostic tests, prescriptions and outcomes.

Secondly, the dataset features **heterogeneous variable types**, encompassing categorical, numerical, identifiers and temporal fields (e.g., dates and timestamps). These characteristics introduce additional complexity, as they require models that can handle mixed-type data while preserving structural and statistical integrity.

Thirdly, each patient sequence contains **contextual attributes** that remain constant

across all events (e.g., gender or date of birth), which must be preserved to maintain logical coherence across time.

Finally, given that each patient record reflects a dynamic sequence of temporally ordered clinical events, it is essential that the selected methodology is capable of capturing both **intra-variable dependencies** (relationships between different variables) and **temporal dynamics** (evolution over time) within each sequence.

In summary, as mentioned in the final point, the generation of high quality synthetic data in this context requires a solution capable of modeling both **inter-variable** and **inter-temporal dependencies** in a consistent manner. The following chapter explores methodologies that meet these requirements and are suitable for this type of data.

4

Methodologies

This chapter presents the methodologies adopted for synthetic data generation throughout this study.

The chapter begins by addressing the need to consolidate the original multi-table clinical dataset into a single, unified tabular format, a requirement imposed by all tested models (sections 4.1 and 4.2). It then describes the technical challenges encountered during preprocessing, including attribute harmonization, table joins and memory-related limitations.

Following the data structuring phase, the chapter delves into the implementation of three distinct synthetic data generation methodologies - `ParSynthesizer`, `ACTGAN` and `DoppelGANger`. For each methodology, specific filtering strategies, configuration parameters and input formatting requirements are described in detail.

Finally, the chapter concludes with an exploration of multi-table modeling using the `HMASynthesizer`, a critical reflection on the limitations observed during experimentation and a brief discussion on alternative modeling paradigms such as Markov Chains, which were considered for their ability to explicitly capture temporal dependencies.

4.1 Initial Data Structuring Considerations

One of the first decisions before applying the methodologies to our data was to define the input structure: whether to preserve the dataset's original multi-table format or to consolidate all information into a single flat table.

During the implementation, it was observed that all methodologies considered in this study require the input data to be in a single-table format and do not natively support multi-table structures. For example, the methodology we tested from SDV[1] library, `ParSynthesizer`, is not designed to handle datasets with multiple related tables and does not support primary-foreign key relationships or hierarchical structures, as the other methodologies we studied in this thesis. This limitation would also be present in all the other methodologies.

[1] https://docs.sdv.dev/sdv/multi-table-data/data-preparation

As a result, it was necessary to merge all relevant tables into a single consolidated dataset. This process involved joining episodes, diagnoses, exams, medications and demographic data into one table that combines all necessary information at the row level.

In the context of clinical episode data, the merging process began by combining the four main episode tables — `Doentes`, `Consultas`, `Urgências` and `Internamentos` — based on the common attribute `DOENTE_ID`. As part of this process, one of the first key harmonization steps was the unification of episode identifiers. Previously, each episode type had its own specific identifier, such as `CON_EPISODIO` for Consultations, `URG_EPISODIO` for Emergencies and `INT_EPISODIO` for Hospitalizations. To address this, a single unified identifier named `CLINICO_EPISODIO` was created. This attribute stores the correct episode ID depending on the type of clinical episode, allowing all subsequent queries and analyses to rely on a standardized episode key. Another important simplification was the creation of the `TIPO_EPISODIO` attribute, which explicitly indicates the episode type. This variable can take one of three values: `'Consulta'`, `'Urgencia'`, or `'Internamento'`. This addition became necessary after merging the episode IDs, as it provides essential context to distinguish between the different types of clinical encounters.

Temporal fields were also harmonized into a single attribute `DTA_EPISODIO`, replacing the original separate attributes such as `DTA_CONSULTA`, `DTA_ADMISSAO` and `DTA_INTERNAMENTO`. The attributes `HR_ADMISSAO` and `HR_INTERNAMENTO` were removed since consultation episodes did not have an associated hour field and introducing artificial timestamps would compromise data integrity.

While an *inner join* was initially considered, it was replaced with an *outer join* to avoid unintentionally excluding patients who did not have records in all episode types. An *inner join* on `DOENTE_ID` would have preserved only those individuals who had events in all three services (consultation, emergency and inpatient), thus significantly reducing the dataset. The use of an *outer join* ensures that patients with partial records are still included in the dataset.

4.2 Combining All Tables into a Unified Dataset

As part of the preprocessing pipeline required for single-table modeling, all relevant tables were merged into a single dataset. This integration included data from the core entities - `Doentes`, `Consultas`, `Internamentos` and `Urgências` - as well as their associated `Exames`, `Análises` and `Diagnósticos`. However, one notable exception was the exclusion of the `Medicamentos_Internamentos` table due to technical constraints. During the join operation, a substantial row multiplication issue was encountered. Each internment episode could already appear in multiple rows within the main dataset due to associations with various analyses, exams or diagnoses. Simultaneously, the `Medicamentos_Internamentos` table contained multiple entries per episode, one for each prescribed or administered drug. As a result, performing a join between the two led to a cross-product effect. For example, an internment episode represented in 10 rows and associated with 5 medications would

result in $10 \times 5 = 50$ rows post-join, in order to represent all possible pairwise combinations between the clinical attributes and the medications.

To avoid this excessive row inflation and maintain dataset manageability, the decision made was to omit the medication component entirely. In addition to the structural issues, this choice was also driven by the limitations of the synthetic data generation methodologies. Including medications would have led to excessive repetition of information across rows, further compromising the quality and coherence of the generated data. As a result, we have a unified dataset composed of 2,286,183 records, excluding `Medicamentos_Internamentos`.

The following subsections describe the main alterations that were necessary to ensure consistency and reduce redundancy after unifying all tables into a single dataset.

4.2.1 Removal of Other Medication Attributes

To ensure structural consistency across all episode types, medication-related attributes from `Consultas` and `Urgências` were also excluded, complementing the removal already applied to `Internamentos`. Although it was technically possible to include medication related attributes for `Consultas` (embedded in the original CSV) and prescription data for `Urgências`, since medications from `Internamentos` were removed from the dataset, it was also decided to remove them from `Urgências` and `Consultas` to maintain uniformity. The removal of medication and prescription related attributes rendered many rows redundant that were previously distinguishable only by those attributes that were removed. As a result, a deduplication step was applied to retain only one unique row per duplicated case. The impact of this deduplication process per entity was as follows:

- **Internamentos:** Initially 1,048,575 rows. After excluding medications, 224,537 redundant rows were removed, resulting in 824,038 unique records.
- **Urgências:** Initially 1,048,575 rows. Following the exclusion of prescription data, 802,008 redundant rows were removed, leaving 246,567 records.
- **Consultas:** Initially 189,033 rows. After removing medication attributes, 88,940 redundant rows were removed, leaving 100,093 records.

After completing this preprocessing and deduplication workflow, the final consolidated table contained a total of **1,170,698 rows**.

4.2.2 Diagnosis Table Simplification

To simplify the handling of diagnosis information, attributes from consultation and internment diagnoses were unified into a common structure, since their codes and descriptions did not present semantic conflicts. Initially, diagnoses were stored separately depending

on the episode type - `Consultas` or `Internamentos` - each with its own associated table. After removing the attribute `TIPO_DIAGNOSTICO`, it became apparent that the diagnostic codes and descriptions used in both contexts did not overlap in a way that would create semantic conflicts (e.g., identical codes with different meanings). Furthermore, since the records from the consultation and internment diagnosis tables did not co-occur for the same episode in the same row, their union was safe and lossless.

As a result, four fields - `COD_DIAGNOSTICO_CONSULTA`, `DES_DIAGNOSTICO_CONSULTA`, `COD_GDH_INT` and `DES_GDH_INT` - were merged into just two general purpose fields: `COD_DIAGNOSTICO` and `DES_DIAGNOSTICO`. This change helped simplify the data structure by reducing the number of attributes, without altering the number of records in the dataset, which remained at 1,170,698 rows.

An additional benefit of this simplification was the reduction in the number of categorical fields, which has a positive effect on the computational efficiency of the synthesis model. High-cardinality categorical variables, especially those processed using one-hot encoding techniques, tend to increase memory consumption and slow down model training. By reducing the number of such variables, the model's performance and scalability were indirectly improved.

4.2.3 Grouping Analysis

The dataset originally contained 288 distinct types of clinical analyses. These were manually grouped into 22 broader categories based on clinical similarity and naming conventions.

The grouping scheme was validated through discussions with multiple healthcare professionals, ensuring that the chosen categories were clinically meaningful and consistent with medical practice. This transformation served two main purposes:

- **Encoding efficiency:** Since this is a categorical column, the reduction in the number of distinct values minimizes the impact of one-hot encoding on memory and model complexity.

- **Pattern learnability:** Fewer, broader categories help the model learn generalized patterns more effectively, whereas the original 288 values were too granular and noisy for meaningful learning.

This transformation was not applied to the `Exames`, despite containing over 500 distinct exam types, because the frequency of exams was significantly lower than that of analyses. The decision was therefore made to prioritize simplification only where the categorical explosion was likely to have the most impact.

4.2.4 Removing Highly Correlated Attributes

An additional analysis was performed to assess pairwise correlations between attributes. Using the reduced dataset (i.e., after removal of prescriptions and medications and following other simplifications), three attribute pairs have near perfect (0.997) or perfect correlation (1.000):

- `COD_DIAGNOSTICO` vs `DES_DIAGNOSTICO`: 1.000
- `COD_ANALISE` vs `DESIGNACAO_ANALISE`: 1.000
- `COD_MED_ESP` vs `ESPECIALIDADE`: 0.997

Based on these results, it was deemed appropriate to retain only the coded identifiers (`COD_DIAGNOSTICO`, `COD_ANALISE`, `COD_MED_ESP`) and eliminate the corresponding descriptive (string) attribute. This simplification reduces the number of high-cardinality categorical columns, while maintaining all relevant information. The slight deviation from 1.000 in the case of `COD_MED_ESP` and `ESPECIALIDADE` was due to a small number of cases where the same `ESPECIALIDADE` label was mapped to different codes. However, each code consistently mapped to a unique specialty, allowing `COD_MED_ESP` to be used as the sole identifier. As such, the attributes `DES_DIAGNOSTICO`, `DESIGNACAO_ANALISE` and `ESPECIALIDADE` were removed from the final dataset.

4.3 Methodologies Implementation

This section presents the practical implementation of the synthetic data generation process using the three selected modeling strategies: `ParSynthesizer`, `ACTGAN` and `DoppelGANger`. Each methodology required distinct data preparation procedures, driven by their architectural constraints. These subsections below describe the filtering criteria, preprocessing logic and configuration details applied to ensure that each model could be properly trained and evaluated under fair and consistent conditions.

4.3.1 Size of the Dataset

An important aspect to consider is the size of the dataset. The dataset used in this study, after the series of preprocessing steps, attribute removals and deduplication procedures, contains 1,170,698 rows, which represents a substantial volume of data. This posed challenges for some of the synthetic data generation methodologies tested, particularly in terms of memory usage and scalability.

- **ParSynthesizer:** `ParSynthesizer`, was only able to process up to approximately 30,000 rows before encountering memory issues. Attempts were made to feed the data in smaller parts (i.e., batching), a strategy commonly supported by deep learning frameworks such as TensorFlow. However, `ParSynthesizer`, while internally based

on PyTorch, does not expose the necessary low level interfaces or support for incremental data loading or batch training. As such, it requires the entire dataset to be loaded into memory from the start, along with full metadata inference and encoding.

Due to `ParSynthesizer`'s full in-memory processing requirements and its limited capacity to handle large datasets efficiently, a filtering strategy was applied during experimentation. Following recommendations from the Colab, the dataset was segmented into two user groups:

- **Group 1 (G1):** Users with 10 or more emergency episodes within any 12-month period between 2016 and 2020.
- **Group 2 (G2):** Users who did not meet the above condition.

Group 1 included approximately 1,110,140 records, while Group 2 contained 60,554 records. Given the substantial size of Group 1, the decision was made to conduct testing using Group 2 instead. As we will see in the following sections, Group 2 was further split to avoid memory overload.

- **DoppelGANger:** In contrast, `DoppelGANger` was capable of processing significantly larger number of rows. During experimentation, it successfully handled 130,000 rows. This scalability is largely due to architectural differences between the two models. While `ParSynthesizer` is based on the `CTGAN` architecture, which requires the entire dataset to fit in memory before training can begin, `DoppelGANger` is designed to train using real data batches. This means it can process large datasets incrementally by dividing them into blocks, making it significantly more memory efficient. `DoppelGANger` is built upon an optimized sequential architecture, such as LSTMs or RNN variants, developed by Gretel.ai with a focus on performance and scalability. Although it requires careful preprocessing (e.g., normalizing date columns), these steps help reduce computational overhead during training.

  Each training sample consists of a unique combination of fixed attributes and a corresponding time series of feature values. The model expects the data to be preprocessed and organized into two separate structures that reflect this distinction:

  > "It supports multi-variate time series (referred to as features) and fixed variables for each time series (attributes)."

  As such, `DoppelGANger` does not natively support multi-table relational databases with foreign key constraints. To apply the model in this context, we had to follow a similar data preparation approach used previously with `ParSynthesizer`. This involved:

  - Joining multiple relevant tables (e.g., patients, exams, medications) into a single unified structure.

  - Structuring the resulting dataset into two partitions: one for fixed attributes and another for time-varying sequences.

  This transformation step was critical to align the real data with the expected input schema of `DoppelGANger` and enable successful model training and generation.

- **ACTGAN:** `ACTGAN` was capable of handling a larger subset of the data. During experimentation, it successfully processed up to 150,000 rows without running into memory or scalability issues. Its architecture, based on CTGAN but optimized within the Gretel framework, enables better memory management than standard SDV models. However, it still requires complete preloading of the dataset and does not support true incremental training. The improved scalability is largely attributed to ACTGAN's simplified tabular modeling (as opposed to sequence-aware generation).

4.3.2 Input Preparation and Configuration

While the input parameters for each model were, in general, kept at their default settings, the data preparation procedures had to be carefully adapted to the architectural requirements of each methodology. The goal was to evaluate the minimum expected quality of the synthetic data that each model could generate under realistic, minimally tuned conditions.

To ensure compatibility and maximize performance, multiple preprocessing adjustments were made according to the constraints and expectations of each model, including sequence structuring, encoding strategies, handling of missing values and date formatting. The following topics describe the specific input transformations applied to each case.

a) PARSynthesizer

- **Sequence Structure Definition:** The attribute `DOENTE_ID` was defined as the *sequence key* and `DTA_EPISODIO` was set as the *sequence index*. This configuration groups all records belonging to the same patient and orders them chronologically by the episode date. The *sequence key* groups rows into sequences and the *sequence index* defines their temporal order within those sequences.

- **Context Columns:** For the generation of sequential data, the following attributes were treated as context: `'DTA_NASCIMENTO'`, `'DTA_OBITO'`, `'GENERO'` and `'CONCELHO'`, as they remain constant throughout each patient's sequence.

- **Date Format Conversion:** All date-related attributes, such as `DTA_EPISODIO`, `DTA_EXAME`, `DTA_ANALISE`, `DTA_NASCIMENTO` and `DTA_OBITO`, were converted from the format `DD-MM-YYYY` to the ISO standard format `YYYY-MM-DD`, as required by the SDV library's date parsers.

- **Handling Empty Strings:** Attributes that were either numeric or categorical could not contain empty strings, as these would cause `ParSynthesizer` to raise errors

during data parsing or metadata inference. All empty string values in such columns were therefore converted to `NaN`, ensuring consistency and compatibility.

b) DoppelGANger

- **Fixed Sequence Length Requirement:** `DoppelGANger` requires all time series sequences to have a fixed length (`max_sequence_len`). This is a technical limitation of the model, it cannot handle sequences of variable length. To comply with this constraint, a fixed maximum sequence length was defined based on the 90th percentile of the number of records per patient. Specifically, the dataset was grouped by `DOENTE_ID` and the number of episodes per patient was calculated.

  The 90th percentile represents the sequence length that covers 90% of patients. For example, if the 90th percentile is 35, this means that 90% of patients have 35 or fewer records. Sequences shorter than this cutoff were padded with zeros, while longer ones were truncated to meet the model's input requirements.

  The 90th percentile was chosen as a compromise between preserving the diversity of the data and ensuring computational efficiency.

- **No Support for Null Values:** As documented in Gretel's official implementation,

  > "DGAN does not support NaNs, please remove NaNs before training."

  This limitation imposes a strict constraint on input preparation: the dataset must not contain any missing values. To address this, two main strategies were considered. The first would involve removing all columns containing missing values, a radical approach that could result in significant information loss. Alternatively, a more balanced solution involves analyzing the percentage of missing data per column and deciding on an appropriate threshold. If the proportion of missing values is low, imputing them with neutral placeholders (e.g., `""` or `0`) is a viable option. Additionally, since GANs can potentially learn that `0` or `""` indicates the absence of information, this encoding does not necessarily degrade model performance. After reviewing the dataset, a threshold of 80% was selected: columns with more than 80% missing values were dropped. This decision was motivated by several factors. Columns with 100% nulls are clearly useless and can be safely removed. For columns with moderate levels of missing data (e.g., 45%, 54%), it is still possible that they contain clinically relevant information. This is particularly true in our dataset, where different episode types, such as consultations, emergencies, or hospitalizations, have distinct sets of relevant attributes. For instance, a consultation episode will not contain values for urgency-related fields, resulting in sparsely populated columns that are still semantically meaningful. Using a more aggressive threshold (e.g., 50%) would risk eliminating such valuable variables, potentially compromising the

quality of the synthetic data. By choosing 80%, we strike a balance between data quality and model robustness.

Following this policy, columns with more than 80% nulls were removed to minimize noise and ensure that sufficient data is available for pattern learning. For the remaining columns, imputation was performed according to data type: categorical columns were filled with empty strings (""), effectively treating missing values as a separate category; numeric columns were filled with zeros, using 0 as a neutral indicator of absence.

- **Date Conversion to Numeric:** `DoppelGANger` does not support timestamps or string-formatted dates. Dates were converted into numeric representations by calculating the number of days since a reference date (e.g., `1900-01-01`). Different base dates were used for different types of dates to prevent overlap and preserve semantic meaning:

```
date_column_bases = {
    "DTA_NASCIMENTO": pd.to_datetime("1900-01-01"),
    "DTA_OBITO": pd.to_datetime("1930-01-01"),
    "DTA_EPISODIO": pd.to_datetime("2000-01-01"),
    "DTA_ALTA_ADMIN": pd.to_datetime("2000-01-01"),
    "DTA_ANALISE": pd.to_datetime("2000-01-01"),
    "DTA_EXAME": pd.to_datetime("2000-01-01")
}
```

  This ensures accurate temporal alignment and avoids encoding issues that would arise from treating timestamps as categorical values.

- **String Handling and Categorical Encoding:** Unlike ParSynthesizer, which internally relies on CTGAN and automatically performs one-hot encoding or embeddings for categorical variables, Gretel's DGAN expects categorical values to be already numerically encoded. `DoppelGANger` cannot directly process raw string values such as `"ALMADA"`, `"MASCULINO"`, or `"NO_COLOR"`, as the model requires all input data to be numerical tensors. Strings lack an inherent vector representation in this context. Therefore, all categorical (string-type) columns were encoded into integer values using `LabelEncoder`. Each encoder was stored individually per column to allow inverse transformation of the synthetic output back into the original string labels after generation. This step is important to ensure that the model can operate on categorical data and that the results can later be interpreted correctly.

  This decoding process is made possible because the categorical encodings were stored in separate files during preprocessing. These mappings are later reloaded and applied to convert the generated numerical outputs back into their original

string representations. The decoding script serves several purposes: (1) it reads the synthetic files generated by DoppelGANger; (2) it reverses the numerical encodings for all categorical features; (3) it reconstructs date fields, using a consistent reference point (e.g., 1900-01-01) and applying `pandas.to_timedelta()` to convert integers back into valid timestamps, while also filtering out values that fall outside an acceptable range (e.g., `[-100000, 100000]`); (4) it reattaches auxiliary fields such as patient identifiers and episode metadata to the generated sequences; (5) it sorts the sequences chronologically; and (6) it outputs the final decoded and human readable synthetic dataset.

- **DOENTE_ID Handling:** The `DOENTE_ID` column does not participate in model training, neither as a sequence index nor as a feature. This is by design: `DOENTE_ID` serves solely as an anchor to group time series during preprocessing and to reconstruct sequences after generation. As a result, all synthetic patients receive artificial identifiers in the format `9000000000 + i`, where `i` is an incremental counter. This behavior explains the lack of variability in this field and guarantees that no real identifiers are memorized or leaked by the model. While it may seem unusual that all synthetic IDs follow a uniform pattern, this outcome confirms that the model does not attempt to replicate or learn from real patient identifiers, preserving privacy.

- **Decoding and Post-Processing:** Once synthetic data was generated, a decoding script was applied with the following objectives:

  1. Read the synthetic files generated by `DoppelGANger`.
  2. Revert encoded categorical values to their original string labels using the encoders saved during preprocessing.
  3. Reconstruct real date values from their numeric representations using a common `base_date` and `pd.to_timedelta()`. The script also handled invalid date ranges (e.g., values outside -100,000 to 100,000 days).
  4. Reassemble sequences with contextual attributes.
  5. Sort and store the final decoded result in a structured and interpretable format.

  This ensures the synthetic output mirrors the original schema and remains human-readable.

c) ACTGAN

- **Lack of Native Sequence Modeling:** `ACTGAN` does not natively support temporal or sequential data structures. Each row is modeled independently, meaning that any temporal or patient-level context, such as multiple episodes for the same individual, is ignored unless manually encoded. As such, if one wishes to preserve some notion of sequence (e.g., episode order or relative timing), it must be introduced explicitly

as additional columns (e.g., sequence index, time deltas). However, the model does not learn true temporal dependencies or inter-row relationships.

- **Fixed Table Format Requirement:** ACTGAN operates on flat, tabular datasets. This means that all input data must be preprocessed into a single table, where each row is a complete and self-contained observation. The model cannot process nested structures, multi-table relationships, or parent-child hierarchies.
- **No Support for Null Values:** Similar to DoppelGANger, ACTGAN does not accept missing values (`NaN`). All nulls must be imputed prior to training. To handle this:
  - Columns with more than 80% missing values were dropped, as these likely lack useful information.
  - For categorical columns, missing values were replaced with empty strings (`""`) or a neutral placeholder such as `"MISSING"`.
  - For numeric columns, missing values were filled with `0`, serving as a neutral indicator of absence.

  This ensures that ACTGAN receives a fully populated dataset, as required for proper model training. Since ACTGAN treats all values literally, missing entries must be encoded in a way that the model can learn to interpret.
- **Categorical Encoding:** ACTGAN uses an internal data transformer that automatically detects and handles categorical variables. However, this behavior depends on correctly defining column data types in the input dataframe. Categorical variables must be of type `object` or `category`, while numerical features must be of type `int` or `float`. Improper typing can lead to incorrect transformations or model errors.
- **Date Handling:** ACTGAN does not natively support date or datetime types. Any date-related columns must be transformed into numeric representations (e.g., number of days since a reference date) or string formats before model ingestion. In this work, date fields were encoded as relative time deltas (e.g., days since `DTA_EPISODIO`) to preserve partial temporal context.
- **DOENTE_ID Handling:** The `DOENTE_ID` was removed from the input before training, as it is a unique identifier that would otherwise cause data leakage and harm generalization. It was later reintroduced into the synthetic dataset using artificially generated patient IDs, similar to the approach used with DoppelGANger.

4.3.3 Modeling Limitations

One major limitation common to all the tested methodologies is the inability to incorporate domain specific rules or enforce logical constraints. These models are fully data-driven and rely entirely on the statistical patterns observed in the training dataset, without any

understanding of semantic correctness or clinical logic. For example, they are unable to ensure that `DATA_OBITO` (date of death) occurs after `DATA_ALTA` (discharge date), or that certain events must follow a specific chronological order such as “prescription after diagnosis” or “exam after consultation.”

None of the models support user-defined rules such as “event X must occur after event Y” or “attribute Z must be null if attribute W is missing.” Consequently, the generated data may include inconsistent or medically implausible sequences, especially if the original dataset contains noise or errors that the model learns and reproduces. This problem is compounded by the lack of model explainability, which limits our ability to trace or correct such issues.

a) PARSynthesizer

Despite its utility for sequential modeling, `ParSynthesizer` exhibits a notable architectural constraint: it only models dependencies at the first level, i.e., between the parent (sequence key) and the immediate child (row-level entries). It is not capable of learning hierarchical or multilevel temporal dependencies such as "child-of-child" (i.e., grandchild) relations. There is currently no way to specify such nested structures or dependency chains within the SDV metadata framework for `ParSynthesizer`. This limitation becomes especially relevant in medical data, where events often exhibit multi-level temporal dependencies. For instance, a patient might undergo a consultation, which leads to a specific exam; the result of this exam then informs the outcome of a follow-up consultation, or perhaps triggers a hospitalization or an emergency visit. Similarly, other downstream events, such as medication prescriptions, additional tests, or even the patient’s eventual death, may all be causally or temporally linked in a nested chain of dependencies. These second order and third order relationships (e.g., exam after consultation, treatment depending on exam results and so forth) cannot be explicitly modeled using `ParSynthesizer`, which flattens such complex dynamics into a simple one-level sequence.

b) ACTGAN

A key structural limitation of `ACTGAN` is its lack of support for temporal sequences. Unlike models designed for sequential modeling, ACTGAN treats each row as an independent, static record, without awareness of chronology or patient history. In clinical datasets, where patients often have multiple related events, this flat representation means the model cannot learn or enforce temporal dependencies such as "exam after consultation" or "diagnosis before treatment." While one might manually include time-related features (e.g., episode indices or time deltas), ACTGAN does not natively interpret or model such sequential relationships. This limitation stems from its design: ACTGAN is essentially a tabular conditional GAN based on CTGAN and thus inherits the same static modeling assumptions.

Another critical limitation is that ACTGAN cannot be trained locally via Python. As of the current version, Gretel has migrated ACTGAN to a cloud only infrastructure, removing access to the model class from the `gretel-synthetics` open-source package. In practice, this means the model can only be executed via the Gretel Console or Cloud API. Unlike other models (e.g., `DoppelGANger`) that can be customized through YAML configuration files or SDK parameters, ACTGAN offers limited control over internal structure or hyperparameters. Features such as `sequence_key`, `context_columns`, or sequence lengths cannot be configured because the model is no longer accessible as editable source code. This lack of transparency and flexibility presents a major barrier to adapt ACTGAN to more complex or domain-specific requirements. Additionally, because training occurs remotely, the model behaves as a black box solution, restricting experiment reproducibility and detailed error analysis.

4.3.4 Adjustments and Experimentations

a) Parsynthesizer - Alternative Hypothesis: Long Format

Initially, the dataset was structured in a wide format, where each row represents a full episode (e.g., consultation or emergency visit) and each column corresponds to a distinct variable. This format results in many null values for attributes that are only relevant in specific contexts (e.g., exams or analyses for consultations but not emergencies), yet ensures that all features are explicitly represented.

Given that some attributes were filled in fewer than 5% of the rows (as observed during the exploratory data analysis), we explored an alternative representation: the *long format*. In this structure, each row corresponds to a single observation of a variable at a specific point in time. Only non null events are preserved, resulting in a more compact and semantically cleaner dataset.

This flattening strategy was used as a practical workaround to simulate "child of child" dependencies within the `ParSynthesizer`, which can only handle a single primary sequence key. The idea was to transform the dataset so that each row represented an atomic clinical event (e.g., consultation, hospitalization, exam, or medication), associated with a unique patient ID (`DOENTE_ID`), a timestamp (`DATA_EVENTO`) and a type label (`TIPO_EVENTO`). All events were then temporally ordered to form a unified timeline per patient.

To preserve the original context in which each event occurred, contextual attributes from the parent episode, such as episode type, medical specialty, or origin, were "broadcasted" to every event row. This ensured that even when events were separated into individual rows, their clinical context remained intact. Compared to the original wide structure, where multiple events coexisted in the same row, this flattened long-format representation provided a cleaner, more structured view of patient histories and allowed the model to better capture temporal dependencies across events. However, this approach quickly proved infeasible. `ParSynthesizer` struggled to process even moderately sized long format datasets and was only able to handle around 12,500 rows before failing due

to memory constraints. Given the original dataset's size and complexity, this limitation rendered the long format approach impractical.

DOENTE_ID	DATA_EVENTO	DTA_NASCIMENTO	GENERO	CONCELHO	KEY	VALUE
9412306383	2020-04-03	1965-01-02	F	ALMADA	TIPO_EPISODIO	Consulta
9412306383	2020-04-03	1965-01-02	F	ALMADA	CLINICO_EPISODIO	9442311815
9412306383	2020-04-03	1965-01-02	F	ALMADA	COD_MED_ESP	20720
9412306383	2020-04-03	1965-01-02	F	ALMADA	ESPECIALIDADE	PSIQUIATRIA
9412306383	2020-04-03	1965-01-02	F	ALMADA	COD_DIAGNOSTICO	NO_CODE
9412306383	2020-04-03	1965-01-02	F	ALMADA	DES_DIAGNOSTICO	UNKNOWN
9412306383	2020-04-07	1965-01-02	F	ALMADA	TIPO_EPISODIO	Consulta
9412306383	2020-04-07	1965-01-02	F	ALMADA	CLINICO_EPISODIO	9442312028
9412306383	2020-04-07	1965-01-02	F	ALMADA	COD_MED_ESP	20720
9412306383	2020-04-07	1965-01-02	F	ALMADA	ESPECIALIDADE	PSIQUIATRIA
9412306383	2020-04-07	1965-01-02	F	ALMADA	COD_DIAGNOSTICO	NO_CODE
9412306383	2020-04-07	1965-01-02	F	ALMADA	DES_DIAGNOSTICO	UNKNOWN
9412306383	2020-04-13	1965-01-02	F	ALMADA	TIPO_EPISODIO	Consulta
9412306383	2020-04-13	1965-01-02	F	ALMADA	CLINICO_EPISODIO	9442310368
9412306383	2020-04-13	1965-01-02	F	ALMADA	COD_MED_ESP	20028
9412306383	2020-04-13	1965-01-02	F	ALMADA	ESPECIALIDADE	ENF. PSIQUIATRIA
9412306383	2020-04-13	1965-01-02	F	ALMADA	COD_DIAGNOSTICO	NO_CODE
9412306383	2020-04-13	1965-01-02	F	ALMADA	DES_DIAGNOSTICO	UNKNOWN

Figure 4.1: Example of the dataset in **long format**.

b) DoppelGANger - Minimal Input Experiment

To further investigate the generalization issues observed in `DoppelGANger`, an experiment was conducted using a simplified version of the dataset. This reduced dataset included only three or four attributes, namely date of birth, date of death and municipality. The objective was to assess whether the modelss performance could be attributed primarily to the high dimensionality of the original input, or if its limitations persisted even with minimal input complexity.

This experiment was motivated by the hypothesis that the low diversity observed in `DoppelGANger`'s output, especially when compared to models like `ParSynthesizer`, might result from the presence of many date fields and complex attribute interactions. By isolating a few critical variables, we intended to test the model's behavior under simplified, low-cardinality conditions.

The results, however, confirmed that `DoppelGANger`'s problems extend beyond input dimensionality. Even with this minimal configuration, the model continued to exhibit severe undergeneralization.

From a test set of 30,000 rows containing **300 unique patients** and **103 unique death dates**, the `ParSynthesizer` produced output with **300 unique patients** and **239 distinct death dates**, a strong sign of generalization, even if some synthetic records did not retain fixed values per patient. In contrast, `DoppelGANger` generated only **72 unique patients** and just **3 unique death dates**, clearly indicating diversity collapse.

These results suggest that the model's weak generalization stems from architectural limitations, which fall into three main categories:

- **Excessive Compression (Underfitting or Over-Regularization):** Even with few attributes, `DoppelGANger` exhibits a sharp reduction in output diversity, particularly in high-cardinality fields such as death dates. Rather than capturing record level variation, the model appears to learn only global patterns and aggregate behavior. This implies that over-regularization may be leading to excessive simplification.
- **Inadequate Modeling of High-Cardinality Fields:** Integer-encoded date fields, such as `DTA_OBITO`, have high dispersion and cardinality. `DoppelGANger` struggles to model these effectively, especially without an inherent temporal structure. It tends to reuse a small set of "safe" values learned during training, explaining the repeated use of the same three death dates.
- **Failure to Preserve Identity Granularity:** Although the model is designed to learn sequential patterns within entities, it fails to enforce diversity across identifiers. As a result, it generated only 72 unique patient IDs from an original set of 300, demonstrating significant duplication and identity collapse.

We can conclude that the poor performance of `DoppelGANger` in this test is not solely due to the complexity of the original dataset. Instead, it appears rooted in the model's structure, its tendency to over-compress, its limitations in handling dispersed continuous fields and its failure to preserve entity level uniqueness.

4.4 Multi-Table Synthetic Data Generation

After running the `ParSynthesizer`, `DGAN` and `ACTGAN` models, primarily focused on individual and sequential tabular data, we decided to try to run a multi-table synthesis model, aiming to determine whether the explicit preservation of table relationships could lead to improvements in synthetic data quality, structural consistency and the diversity of generated records.

To this end, we selected a model from the `SDV` (Synthetic Data Vault) library, which offers multiple approaches for multi-table synthetic data generation. Among the available options, `Day Z Synthesizer`, `Independent Synthesizer`, `HSA Synthesizer` and `HMA Synthesizer`, we opted for the **HMA Synthesizer**.

The `HMA Synthesizer` is the most comprehensive model available in the open-source version of the library and it is designed to capture hierarchical relationships between tables using real data. Despite some limitations (such as the maximum number of supported tables and limited relational depth), the HMA model offers a balanced compromise between structural integrity and synthetic data quality, making it the most suitable choice for the objectives of our experimentation. During the construction of the synthetic data pipeline, we encountered a schema limitation in the original hospital database schema: several core tables, such as `Consultations`, `Emergencies` and `Hospitalizations`, did not expose a single-column surrogate key. This occurs because a single clinical episode

(e.g., CON_EPISODE, URG_EPISODE) can generate multiple related entries, such as several prescriptions or medical exams linked to the same event.

However, for training the HMASynthesizer, each table is required to have a unique primary key to ensure internal referential mapping. To address this, we created artificial technical keys (e.g., SEQ_CONSULTATION_ID, SEQ_URG_ID) to serve as primary keys.

4.4.1 Data Preparation and Modeling Pipeline

The modeling process involved several critical data engineering steps to ensure compatibility with the requirements of the HMASynthesizer. First, the necessary tables, including Patients, Consultations, Emergencies and Hospitalizations, were loaded from CSV files. To try to reduce sparsity and improve modeling accuracy, large mixed tables (such as Consultations, which contained embedded information on diagnostics, exams and prescriptions) were normalized and split into nine logically independent tables. Referential integrity was enforced by validating that all foreign key references (e.g., PATIENT_ID) were consistent across related tables.

All date columns were converted to the datetime64 format to support correct temporal interpretation during modeling. Since many of the original tables did not contain truly unique primary keys, often due to multiple records being associated with a single clinical episode, artificial identifiers (e.g., SEQ_*) were generated to ensure row-level uniqueness. At the same time, the original episode identifiers (e.g., CON_EPISODE, URG_EPISODE) were retained as standard id-type fields to preserve semantic meaning and support relationship definitions.

The overall schema and table relationships were formalized using the MultiTableMetadata object, where primary and foreign keys were explicitly defined and relationships were established using the add_relationship method. Once the metadata and real data were prepared, the HMASynthesizer was trained to learn the statistical distributions and hierarchical dependencies between tables.

Finally, the synthetic data was exported to individual CSV files per table, each named following the pattern MultiTable Output - <TableName>.csv, enabling subsequent analysis or reuse in downstream tasks.

5

Results Evaluation

This chapter aims to provide a comprehensive evaluation of the models developed during the experimentation process, focusing on three key aspects: Integrity, Privacy and Utility. In addition, a manual inspection pipeline is explored to compensate for the limitations of the available automated metrics.

The chapter concludes with a discussion of the obtained results, highlighting not only the strengths of each model but also the issues encountered throughout the process that negatively impacted their overall performance and evaluation.

5.1 Evaluation Process

After the initial training phase, all three methodologies were first executed using a filtered dataset containing approximately 30,000 rows. This value was chosen to ensure compatibility with `ParSynthesizer`, which was the most memory constrained model and unable to handle larger volumes. To enable a fair comparison across all models, the same input subset was used for `ACTGAN` and `DoppelGANger` in the first evaluation stage.

However, since both `ACTGAN` and `DoppelGANger` are capable of handling larger datasets, additional runs were performed using a significantly larger input, 130,000 rows, to assess whether increasing the training volume would improve the quality of the generated data. This step allowed for a more realistic evaluation of these models under their full potential and helped identify whether the limitations observed in smaller samples were related to the reduced dataset size or to structural constraints inherent to the models themselves.

The Gretel library offers powerful tools for evaluating the integrity, utility and privacy of synthetic data, including `evaluate_data_quality`, `evaluate_data_privacy` and `evaluate_ml_quality`. However, these functions are only available through Gretel's cloud-based platform and are not included in the publicly available `gretel-synthetics` Python package.

In our study, while `ACTGAN` was executed via Gretel's cloud infrastructure, other models such as `ParSynthesizer` and `DoppelGANger` were run entirely in a local environment.

Because Gretel's evaluation functions are designed to operate natively within their platform, we were unable to apply them to synthetic data generated locally or outside the prescribed workflows. Furthermore, Gretel does not currently support importing externally generated synthetic datasets into its evaluation dashboard, limiting consistent metric computation across all models. To overcome this limitation and ensure fair comparison, we relied on open-source tools such as `pandas`, `scikit-learn`, `scipy` and `sdmetrics`. These tools were used to approximate, as closely as possible, the evaluation dimensions provided by Gretel's proprietary functions - namely statistical integrity, privacy risk and downstream machine learning utility.

5.2 Empirical Evaluation Pipeline

Recognizing the inherent weaknesses and limitations of existing synthetic data evaluation metrics, we decided to adopt a "human in the loop" methodology to critically assess the quality of synthetic datasets besides the application of automated metrics.

This manual evaluation process plays a crucial role in detecting obvious logical inconsistencies and validating the overall plausibility of the generated data, especially when temporal order and per-entity coherence are critical, as is the case in clinical datasets.

Although widely used, existing metrics are not fundamentally flawed, but rather limited in scope, they capture specific aspects of data quality while overlooking others.

We implemented an evaluation script that automates the metric computation, rule-based checks and report generation described above. The complete listing is provided in annex III. The script is deliberately general-purpose and can be readily adapted to any dataset or schema by adjusting parameters. A public, data-free version is also available on GitHub.[1]

5.2.1 Manual Pre-Metric Verification Procedure

Before applying any metrics, we conducted a verification process aimed at confirming whether the synthetic data respected basic rules of integrity and logical consistency. The checks focused particularly on temporal plausibility and attribute stability. The following validations were performed manually by directly inspecting the data:

- **Per-Patient Temporal Coherence:** Ensured that patient-level attributes such as `DTA_NASCIMENTO`, `DTA_OBITO`, gender and municipality remained constant across all records associated with the same `DOENTE_ID`.
- **Life Logic Validation:** Confirmed that death dates, when present, occurred after any clinical events (exams, consultations, emergency, hospitalizations, ...). Any record with an event following a death was flagged as logically invalid.

[1] https://github.com/Abreu6/Empirical_Evaluation_Script

- **Death Date Distribution:** Analyzed the distribution of `DTA_OBITO` to check for suspicious patterns, anomalies, or excessive repetition.

- **Attribute Distribution Comparison:** Compared the distribution of key attributes (e.g. diagnosis codes) between real and synthetic datasets.

- **Temporal Sequencing of Episodes:** Validated that dependent events such as lab tests (`DTA_ANALISE`) or medical exams (`DTA_EXAME`) occurred only after the start of the corresponding episode (`DTA_EPISODIO`). Occurrences where these procedures took place before the patient's admission were flagged as illogical and undesirable, as they violate basic temporal consistency.

- **Hospital Discharge Logic:** Checked that discharge dates (`DTA_ALTA_ADMIN`) occurred after admission dates (`DTA_EPISODIO`), except in cases where discharge was missing.

- **Contextual Presence of Triage Attributes:** Confirmed that triage-related attributes (`COR_TRIAGEM`, `QUADRO_TRIAGEM`) appeared only in emergency episodes and vice versa.

To ensure the consistency of the evaluation, we also validated that these rules were respected by the real dataset. A data cleaning step was applied to remove any logically inconsistent records from the real data.

During this process, we identified and removed 139 records out of 1,170,694 in which clinical events occurred after the recorded date of death, violating the life logic rule. Figure 5.1 illustrates the SQL query used to identify and isolate the 139 inconsistent records with post-mortem clinical events. Additionally, 1,702 records were found where lab analyses (`DTA_ANALISE`) or medical exams (`DTA_EXAME`) were timestamped before the official start of the corresponding clinical episode (`DTA_EPISODIO`). Such sequences are clinically implausible and indicate a lack of temporal integrity.

By eliminating these inconsistencies, we ensured that the real dataset adhered strictly to the same logic-based constraints later applied to the synthetic data, thereby establishing a valid baseline for comparative evaluation.

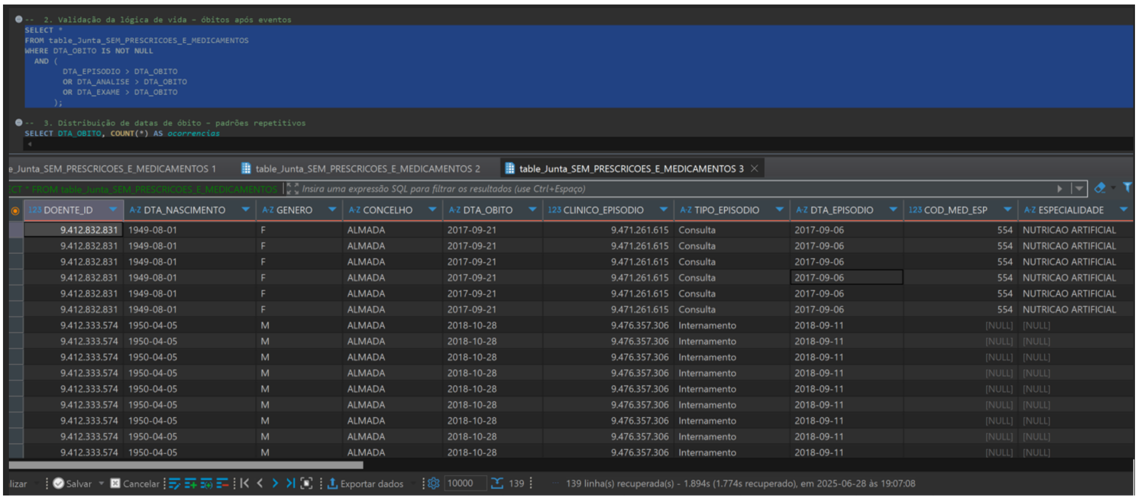


Figure 5.1: SQL query used to detect the 139 records with clinical events occurring after death.

5.2.2 Experimental Evaluation Strategy

To determine the specific point at which each synthetic data generation model (`ParSynthesizer`, `ACTGAN` and `DoppelGANger`) begins to exhibit logical or temporal inconsistencies, we designed an incremental evaluation framework based on varying levels of dataset complexity.

Evaluation Levels of Structural Complexity

1. **Level 1 - Population Only:** Includes only the `Doentes` (Patients) table.
2. **Level 2 - Population + Consultations:** Adds the `Consultas` table. Equivalent complexity could also be reached by adding `Internamentos` or `Urgências`.
3. **Level 3 - Population + Consultations + Emergency Visits:** Integrates multiple episode types to test handling of overlapping clinical contexts.
4. **Level 4 - All Episodes + Exams + Analyses:** Incorporates all major episode types and their associated `Exames` and `Análises`, increasing sequential and logical complexity.
5. **Level 5 - Full Dataset:** Includes all episode types, diagnoses, medications, prescriptions and supporting entities.

Evaluation Process at Each Level

For every level of complexity, a systematic validation process was applied. The previously defined manual validation pipeline was used to assess the logical, temporal and semantic consistency of the synthetic data. Particular attention was given to identifying when each model began to violate the established data rules such as generating events after a patient's recorded death, producing exams before an episode start date, or introducing inconsistencies in patient-level attributes. This manual review formed the foundation for

a comparative analysis aimed at evaluating each model's robustness and generalization capacity. Only after completing this qualitative assessment were standard synthetic data quality metrics applied, including SDMetrics outputs, visual inspections and distribution stability measurements.

5.2.3 Comparative Evaluation by Complexity Level

The evaluation of the synthetic data generated by each model was conducted across increasing levels of complexity to determine how far each model can maintain coherence before breaking down. The process begins with only the patient table and progressively incorporates additional relationships and tables, such as consultations, emergency visits, exams, lab analyses, medications and prescriptions. This step by step approach allows us to pinpoint the exact stage at which each model begins to fail and understand the nature of those failures.

Level 1 - Population Only (`Doentes`)

At this level, the focus is on validating core demographic information. This includes ensuring that the number of patients generated is consistent with the original distribution, that each `DOENTE_ID` maintains a stable date of birth and that attributes such as `GENERO` and `CONCELHO` are internally coherent. Additionally, we check the frequency and distribution of `DTA_OBITO` values and verify whether the number of deceased patients is realistic.

- **ParSynthesizer:** Strong performance with consistent attributes like date of birth, gender and municipality across patients. No significant issues.
- **ACTGAN:** Excellent generalization of population structure. Outputs are diverse and free of logical inconsistencies.
- **DoppelGANger:** Signs of overfitting emerge early. Some repeated `DOENTE_ID` values show inconsistent birthdates. Weak distributional generalization.

Level 2 - Population + Consultations

This level introduces temporal and relational dependencies between patients and consultation episodes. Key validation points include ensuring that consultations occur after the patient's birth and, crucially, before their date of death. We also assess whether each consultation is correctly linked to an existing patient and whether patient attributes remain stable across related episodes.

- **ParSynthesizer:** Maintains good coherence between patient and consultation episodes. Minor date drift, but no major logical violations. Some consultations after death.
- **ACTGAN:** Still performs robustly. Consultations appear on plausible dates and are correctly linked to patients.

- **DoppelGANger:** Issues become more pronounced. Consultations sometimes occur before birth or after death, with unstable associations to patients.

Level 3 - Population + Consultations + Emergency Visits

- **ParSynthesizer:** Starts to introduce logical errors. Emergency specific attributes (e.g., triage) appear incorrectly in consultations. Confusion between episode types.
- **ACTGAN:** Continues to perform well. Emergency episodes are correctly structured and triage attributes are used appropriately.
- **DoppelGANger:** Severe failure. Attributes from different episode types mix. Events occur out of chronological order (e.g., before birth, after death).

Level 4 - All Episodes + Exams + Analyses

- **ParSynthesizer:** Quality deteriorates. Exams and analyses sometimes occur before episode start, breaking sequential logic.
- **ACTGAN:** Minor inconsistencies emerge, but less than 1% of exams are out of order. Overall structure remains plausible.
- **DoppelGANger:** Becomes unusable. Exams occur before birth, multiple dates of death per patient and repeated episodes.

Level 5 - Full Dataset (Medications + Prescriptions)

- **ParSynthesizer:** Fails in therapeutic structure. Prescriptions and administrations are mismatched or missing.
- **ACTGAN:** Begins to struggle, but maintains the best performance among all models. Some relational inconsistencies appear.
- **DoppelGANger:** Completely breaks down. Medications appear without context, prescriptions are unlinked or assigned to the wrong patients. Total loss of clinical logic.

5.2.4 Generalized Synthetic Data Evaluation Pipeline

The goal is to ensure that generated data retain essential properties such as logical consistency, statistical realism and temporal plausibility. The aspects listed below represent generic criteria that any evaluation framework ought to verify in order to guarantee these characteristics.

- **Attribute Stability per Identifier:** Attributes such as date of birth, gender, or municipality should remain constant across all records sharing the same identifier. Multiple values per ID indicate inconsistency.

- **Temporal Sequencing Checks:** The temporal relationship between clinical events must follow a logical order. For example, hospital discharge dates (`DTA_ALTA_ADMIN`) should always occur after the corresponding admission date (`DTA_EPISODIO`) and procedures such as lab analyses (`DTA_ANALISE`) or medical exams (`DTA_EXAME`) must not precede the clinical episode they belong to. Additionally, when a death date (`DTA_OBITO`) is present, it must always follow all other clinical events for that patient.

  Beyond simple pairwise comparisons, it would be desirable to define a set of attributes as temporal components and specify a *partial order* among them. Such a partial order can be represented as a directed acyclic graph (or tree) and the evaluation framework should verify that all records respect this ordering. This would allow more flexible and comprehensive sequencing checks across complex event structures.

- **Distribution Alignment:** Check that continuous or temporal variables (e.g., episode duration, age, dates) in synthetic data match the real data in distribution. Use:

  - Histograms, boxplots, KDE plots;
  - Statistical descriptors: mean, std, skewness, kurtosis;
  - Statistical distances: Jensen-Shannon divergence, Wasserstein distance.

  While numerical summaries and distance measures are informative, they often fail to capture differences in distributional shape. A canonical example is Anscombe's Quartet [45]: four datasets with nearly identical means, variances and correlations, yet clearly distinct when plotted. This illustrates why it is essential to supplement quantitative measures with visual diagnostics such as histograms, boxplots, so that discrepancies unnoticed by summary statistics become visually evident.

- **Contextual Co-occurrence Rules:** Some attributes should only appear under certain conditions. For example, triage variables should only appear in emergency episodes. Violations of this logic signal contextual inconsistency. More generally, these rules can be expressed either as *logical implications* (e.g., "if an episode is not of type Emergency, then all triage-related fields must be missing") or as *dependence constraints* between variables (e.g., two variables that are known not to be independent should preserve their statistical association in the synthetic data). In this way, contextual rules cover both deterministic co-occurrence patterns and probabilistic dependencies that ought to be respected.

- **Realistic Categorical Frequencies:** Categorical variables should preserve plausible frequency distributions. It is useful to distinguish between *nominal* categories (purely qualitative, such as gender or municipality) and *ordinal* categories (with an inherent order, such as triage color or disease severity levels). While nominal categories are evaluated mainly by their relative proportions, ordinal categories require additional

checks to ensure that the natural ordering is respected and that the synthetic data reproduces the expected gradients across levels.

Building upon the manual verification procedure, we propose a generalized evaluation pipeline that can be applied to any synthetic dataset, with the goal of ensuring that generated data preserve essential properties of logical consistency, statistical realism and temporal plausibility. In practice, these principles were implemented through a Python script, presented in Annex III, where each of the general criteria is reflected in a concrete step of the pipeline:

1. **Data loading and normalization:** at the beginning of the script, the real and synthetic datasets are imported, text encodings are harmonized (accents, case, whitespace) and all date fields are parsed into a common format.

2. **Derived attributes:** immediately after loading, additional variables are created, such as patient age at the time of episode and the episode duration, both obtained from differences between date fields.

3. **Temporal rules:** in a subsequent step of the script, explicit comparisons between dates enforce ordering constraints (e.g., admission before discharge, episode before exam or analysis, birth before death).

4. **Distribution alignment:** further on, the distributions of continuous variables are compared using Kolmogorov–Smirnov tests and descriptive statistics, while categorical variables are assessed through frequency distributions and χ^2 tests.

5. **Static consistency:** the script then groups records by identifier and verifies that attributes expected to remain constant (e.g., gender, birth date) are not altered across multiple records of the same individual.

6. **Anomaly scans:** towards the end, dedicated routines check for repeated or unlikely death dates and flag negative or implausible values in the derived attributes.

Together, these steps provide a reusable framework to validate logical consistency, statistical realism and temporal plausibility in synthetic datasets.

5.3 Results

The following metrics were computed on synthetic datasets of 30,000 records, ensuring that all models were evaluated under identical conditions and are directly comparable.

5.3.1 Integrity Metrics

To evaluate the practical integrity of the generated synthetic data, integrity metrics were computed for each model, as presented in Table 5.1. These include the *Field Correlation Stability*, which assesses the consistency of correlation matrices between real and synthetic data; *Deep Structure Stability*, designed to evaluate high-order feature interactions; and *Field Distribution Stability*, which measures how closely individual feature distributions are maintained. From these, the *Synthetic Quality Score (SQS)* is calculated as a weighted combination of the three, providing an overall indicator of integrity. Additionally, the MLQS reflects the ability of machine learning models trained on synthetic data to replicate performance achieved on real data.

Table 5.1: Integrity Metrics Comparison

Metric	**ParSynthesizer**	**DoppelGANger**	**ACTGAN**
Field Correlation Stability	0.8864	0.7277	0.9538
Deep Structure Stability	0.0000	0.0000	0.7100
Field Distribution Stability	0.4832	0.6867	0.8054
Synthetic Quality Score (SQS)	0.4565	0.4715	0.5915
ML Quality Score (MLQS)	1.0000	0.2899	1.0000

Among the results, `ACTGAN` achieved the highest SQS (0.5915), with strong performance in field distribution (0.8054) and correlation stability (0.9538), indicating that its outputs are both statistically coherent and suitable for downstream tasks. Notably, it was the only model to record a non-zero *Deep Structure Stability* (0.7100), suggesting some capacity to capture complex multivariate dependencies. `ParSynthesizer` and `DoppelGANger` both scored 0.0000 in this metric, reflecting a failure to model deeper feature interactions. Interestingly, `ParSynthesizer` and `ACTGAN` both achieved perfect ML Quality Scores (1.0000), which could indicate potential overfitting or memorization of patterns from the original data. The consistent absence of deep structure stability in two of the three models further reinforces concerns about their generalization capabilities and depth of learning.

5.3.2 Utility Metrics

To assess how well each model preserved the original statistical structure of the training data, a set of utility metrics was computed, as presented in Table 5.2. These include the *Column Shapes Score*, which evaluates how closely the univariate distributions of each feature were replicated; the *Column Pair Trends Score*, which measures pairwise relationships between columns; and the *Overall Score*, which represents the average of the

two. Additional measures such as *Correlation Similarity*, *Statistic Similarity* and *Boundary Adherence* provide further insight into multivariate dependencies, statistical moments and whether generated values fall within plausible boundaries.

Table 5.2: Utility Metrics Comparison

Metric	**ParSynthesizer**	**DoppelGANger**	**ACTGAN**
Column Shapes Score	70.12%	43.37%	83.42%
Column Pair Trends Score	61.94%	51.48%	74.52%
Overall Score (Average)	66.03%	47.42%	78.97%
Correlation Similarity	0.9036	0.8299	0.9428
Statistic Similarity	0.8644	0.6133	0.9688
Boundary Adherence	0.8804	0.4949	1.000

Among the tested models, `ACTGAN` consistently outperformed the others across all utility indicators, achieving an overall score of 78.97% and perfect boundary adherence. `ParSynthesizer` showed moderate utility, preserving correlations reasonably well (0.9036), but with lower pairwise trend alignment. In contrast, `DoppelGANger` recorded the weakest utility performance, particularly in distributional alignment and boundary respect, indicating greater deviation from the structure of the real data.

5.3.3 Privacy Metrics

To evaluate the extent to which the synthetic data preserves patient confidentiality and mitigates disclosure risks, a set of privacy-related metrics was computed using the SDMetrics library. As shown in Table 5.3, these include the *Membership Inference Score*, which assesses the risk of determining whether a given record was part of the original dataset; the *Attribute Inference Score*, which estimates how easily sensitive attributes can be inferred from known values; the *PII Replay Score*, which measures the degree to which personally identifiable information (PII) is replicated in the synthetic data; and the *Disclosure Protection* score, which combines various privacy aspects into a single indicator.

Table 5.3: Privacy Metrics Comparison

Metric	**ParSynthesizer**	**DoppelGANger**	**ACTGAN**
Membership Inference Score	0.0000	0.0000	0.0000
Attribute Inference Score	1.0000	0.2595	0.4497
PII Replay Score	1.0000	1.0000	0.9100
Disclosure Protection	0.8258	0.7270	0.9200

Note that for the *PII Replay Score*, the ideal value is **0.0**, as higher scores indicate stronger replication of personally identifiable information (undesirable).

All three models recorded a Membership Inference Score of 0.0000, indicating strong resistance to membership inference attacks under the evaluation configuration. However,

considerable variation was observed in the other metrics. `ParSynthesizer` presented perfect scores (1.0000) in both PII Replay and Attribute Inference, raising significant concerns about potential memorization or leakage of sensitive data. Since the ideal PII Replay Score is **0.0** (indicating no personally identifiable information is reproduced), a perfect 1.0000 actually reflects the worst-case scenario, where sensitive values are fully replicated in the synthetic dataset. Although its Disclosure Protection score (0.8258) was relatively high, these individual indicators suggest a fragile privacy profile. On the other hand, `DoppelGANger` and `ACTGAN` offered a more balanced trade-off: both models showed substantially lower Attribute Inference Scores (0.2595 and 0.4497, respectively), with `ACTGAN` also achieving the highest overall Disclosure Protection (0.9200). While `DoppelGANger` maintained perfect PII Replay, this further raises concerns given its lower synthesis diversity noted in other evaluations. These findings reinforce the importance of not evaluating privacy in isolation but rather in conjunction with integrity and utility, as overly strict privacy controls (as seen in `ParSynthesizer`) may come at the cost of degraded data quality.

5.3.4 Additional Assessment: More Runs Comparison

To explore the impact of dataset size on model performance, an additional run of `DoppelGANger` was conducted with 130,000 records using the same preprocessing pipeline. This run is considered the model's best-case configuration in terms of resource utilization and batch setup. A similar large-scale run was performed with `ACTGAN` to compare scalability and performance consistency. The results reveal a nuanced trade-off: while both models showed improvements in several utility and integrity metrics with more data, some privacy-related scores (such as PII Replay and Attribute Inference) remained high, suggesting persistent leakage or overfitting tendencies.

Table 5.4: 130,000 Run Metrics Comparison: DoppelGANger vs ACTGAN

Metric	**DoppelGANger**	**ACTGAN**
Field Correlation Stability	0.5827	0.8906
Deep Structure Stability	0.0000	0.5021
Field Distribution Stability	0.6878	0.8054
Synthetic Quality Score (SQS)	0.4235	0.5927
Membership Inference Score	0.0000	0.0000
Attribute Inference Score	0.3827	0.2563
PII Replay Score	1.0000	0.6650
Machine Learning Quality Score	0.3659	0.9418
Column Shapes Score	45.86%	85.47%
Column Pair Trends Score	34.74%	71.56%
Overall Score (Average)	40.3%	78.51%
Correlation Similarity	0.7962	0.8906
Statistic Similarity	0.6233	0.9888
Boundary Adherence	0.5730	1.0000

While both models achieve nearly identical metrics at the 130,000 scale, it is important to emphasize that for fairness in evaluation across all models, the 30,000 sample runs remain the reference in the main results tables.

To conclude the evaluation, a final comparison was performed between the baseline run of `ParSynthesizer` (30,000 records) and the best large-scale configurations of `DoppelGANger` (130,000 records) and `ACTGAN` (150,000 records). This table highlights how increased dataset sizes affected performance across models when contrasted with the smaller-scale baseline.

Table 5.5: Final Comparison: ParSynthesizer (30k) vs. DoppelGANger (130k) vs. ACTGAN (150k)

Metric	**ParSynthesizer**	**DoppelGANger**	**ACTGAN**
Field Correlation Stability	0.8864	0.5827	0.8856
Deep Structure Stability	0.0000	0.0000	0.6500
Field Distribution Stability	0.4832	0.6878	0.8054
Synthetic Quality Score (SQS)	0.4565	0.4235	0.6013
Membership Inference Score	0.0000	0.0000	0.0000
Attribute Inference Score	1.0000	0.3827	0.3563
PII Replay Score	1.0000	1.0000	0.6750
Machine Learning Quality Score	1.0000	0.3659	0.9418
Column Shapes Score	70.12%	45.86%	82.62%
Column Pair Trends Score	61.94%	34.74%	70.19%
Overall Score (Average)	66.03%	40.3%	78.51%
Correlation Similarity	0.9036	0.7962	0.8906
Statistic Similarity	0.8644	0.6233	0.9288
Boundary Adherence	0.8804	0.5730	1.0000

5.4 Discussion

a) PARSynthesizer

`ParSynthesizer` revealed multiple critical limitations in the context of this study, particularly its inability to support constraints during the synthetic data generation process. This limitation significantly compromises the practical utility of the model, especially in scenarios where logical consistency, valid domain enforcement, or attribute coherence is essential. Although some evaluation metrics yielded minimally acceptable results, a qualitative analysis of the synthetic data uncovered multiple inconsistencies that undermine the reliability of the generated dataset. One prominent issue was the generation of clinical events occurring after a patient's date of death, an error that could theoretically be prevented through the use of logical constraints. However, `ParSynthesizer` does not support the specification of such rules, making it impossible to enforce basic real world conditions in the synthetic outputs. Furthermore, the model exhibited difficulties in capturing complex temporal and hierarchical relationships, such as those involving nested dependencies

(e.g., "child-of-child" structures). Without the ability to accurately model event sequences or maintain temporal coherence between related records, `ParSynthesizer` fails to preserve essential structural aspects of the original dataset. Unlike other synthesizers available within the same SDV framework `ParSynthesizer` does not provide any interface to define explicit constraints. This absence, combined with the model's inability to infer implicit rules from data, results in the generation of synthetic records that are often logically or temporally invalid.

Analysis of Evaluation Metrics
The metrics obtained reinforce these limitations:

- **Moderate Fidelity:** While correlation similarity and boundary adherence were relatively high, the overall fidelity score indicates that the generated data lacks completeness and structure, especially in multivariate aspects.
- **Limited Utility:** Despite a good Field Correlation Stability score (0.8864), both Deep Structure Stability (0.0000) and SQS (0.4565) reveal that the model does not effectively capture meaningful latent relationships across features.
- **Privacy Concerns:** Although the membership inference score was low (0.0000), which is desirable, the attribute inference and PII replay scores were both 1.0000, indicating potential data leakage or overfitting.
- **Model Overconfidence:** The Machine Learning Quality Score (1.0000) was unusually high. While this may seem positive, such a perfect result is highly suspicious and typically signals a lack of generalization, possibly due to data memorization.
- **Observed Limitations:** In practice, the synthetic data displayed low diversity in categorical variables and failed to maintain logical and temporal constraints between columns, further undermining its applicability in realistic clinical scenarios.

In conclusion, `ParSynthesizer`, in its current form, is not a suitable tool for synthetic data generation in domains where strong logical and temporal dependencies must be preserved. While it offers potential for sequence-based modeling in simpler contexts, its lack of constraint handling, limited capacity for capturing deep relational structures and difficulty in scaling to high-dimensional datasets with larger record volumes make it inadequate for the requirements of this study.

b) DoppelGANger

`DoppelGANger`, despite being designed specifically for time-series generation, demonstrated severe limitations in terms of generalization and privacy when evaluated on the clinical dataset used in this study. The model's performance was dominated by signs of overfitting. In multiple runs, it exhibited perfect scores in *Attribute Inference Score* and *PII*

Replay Score (1.0000), which strongly indicates that the model memorized and replicated portions of the real dataset. This undermines the very objective of synthetic data generation, privacy preservation and generalization. Furthermore, `DoppelGANger` struggled with maintaining statistical integrity. In certain runs, metrics such as *Field Distribution Stability* (0.4832) and the *Synthetic Quality Score* (0.4565) were notably low, revealing the model's inability to reproduce the statistical properties of the real data accurately. While the MLQS reached a perfect 1.0000, this result, in context, is misleading. Combined with high leakage and inference scores, such an MLQS indicates that downstream models trained on synthetic data perform well only because the data is effectively a copy of the real dataset, not because it captures generalizable patterns. Thus, the high MLQS is symptomatic of memorization rather than a sign of quality. Another important limitation was observed in the *Deep Structure Stability* metric, which consistently returned a score of 0.0000. This suggests the model is unable to capture deeper latent structures or preserve high-level relationships across dimensions, an essential requirement when modeling hierarchical or structured clinical data. Finally, a qualitative inspection of the generated data revealed a severe lack of variability, with sensitive attributes such as `DTA_OBITO` taking on only two unique values across the entire synthetic dataset. This further reinforces the diagnosis of overfitting and limited capacity for abstraction. In addition to these metric-based observations, several structural and practical limitations became apparent during implementation. The model requires all sequences to have the same length, which necessitated preprocessing steps such as truncation and padding. These operations, while technically required, distorted the natural clinical trajectories and reduced the temporal fidelity of the sequences. Moreover, `DoppelGANger` does not support missing values, a common and semantically important feature in clinical datasets. To adapt the data, informative columns had to be removed or null values were replaced with artificial placeholders (e.g., empty strings or zeros), reducing the model's capacity to learn realistic patterns, especially where the absence of information is itself meaningful (e.g., no exam requested).

Critically, the model also lacks any support for logical constraints. As a result, multiple temporal and semantic inconsistencies were identified in the synthetic data. Examples include:

- Records missing key fields such as `DTA_EPISODIO`;
- Clinical events (`DTA_ALTA`, `DTA_ANALISE`) occurring before the start of an episode;
- Episodes taking place after the recorded date of death (`DTA_OBITO`);
- Exams registered before a patient's birth date or before the associated episode.

These logical violations are visualized in Table 5.6, which quantifies the most frequent inconsistencies found:

Table 5.6: Detected Errors and Inconsistencies in the Synthetic Data

Type of Error	Affected Records
Death date before date of birth	19,329
Episode date before date of birth	22,310
Exams recorded after hospital discharge	6,860
Laboratory analyses after discharge	8,107

Comparison to ParSynthesizer.
`DoppelGANger` was compared directly to `ParSynthesizer`, revealing the following weaknesses:

1. **Lower Data Diversity:** Fewer unique values in exams, identifiers and date fields.
2. **Poorer Temporal Quality:** Unrealistic clustering of critical dates like `DTA_OBITO`.
3. **Flat Distributions:** Failure to capture the complexity and heterogeneity of the real dataset.
4. **Higher Null Rates:** Reduced realism in the presence of key medical events or outcomes.

In sum, `DoppelGANger`, in the evaluated configuration, fails to generalize beyond the training data. Its synthetic outputs are marked by low statistical diversity and high privacy risk, making it unsuitable for applications that demand generalization and secure data synthesis. Compared to `ACTGAN` and `ParSynthesizer`, this model presented the weakest balance between utility and privacy. Additionally, the structural constraints imposed by its architecture, namely fixed-length sequences, no support for missing values and the absence of logical constraints, further compromise its usefulness in real-world clinical scenarios.

c) ACTGAN

Similar to `ParSynthesizer`, `ACTGAN` does not maintain consistent values for certain static attributes across records belonging to the same patient, most notably `DTA_OBITO`. This behavior is expected, given that the model is not sequential and does not incorporate temporal logic during training. As a result, temporal inconsistencies can still emerge in the generated dataset.

The overall performance of `ACTGAN` was positive, particularly in terms of statistical fidelity. Both output samples (30,000 and 150,000 records) demonstrate that the model was able to replicate the structure of the original clinical dataset with high integrity. The outputs are well-formatted, exhibiting a clear separation between static patient-level attributes (e.g., birth date, gender, municipality) and dynamic episode-level variables (e.g., lab analyses, diagnoses and exams).

One of the most notable strengths of ACTGAN is the diversity of generated values in categorical columns such as COD_ANALISE, COD_ACTO_MEDICO and DESIGNACAO_EXAME. These results suggest that the model effectively learned the underlying distributions and relationships in high-cardinality columns, in stark contrast to DoppelGANger, which often resorted to repeated generic placeholders.

Quantitatively, ACTGAN achieved high scores across various statistical metrics. For example, the Column Shapes Score and Column Pair Trends Score surpassed 83%, indicating strong alignment with the univariate and bivariate distributions of the original dataset. The Boundary Adherence metric reached 1.0, confirming that all generated numerical values remained within the minimum and maximum observed in the real data. These results reflect the model's architectural advantages, including the use of tabular attention mechanisms, conditional training on discrete variables and a robust adversarial setup that encourages marginal and joint distribution alignment.

However, these statistical strengths do not fully translate to semantic or clinical consistency. The most critical limitation observed was a severe collapse in diagnostic code variability. More than 90% of records generated by ACTGAN contained the placeholder value "NO_CODE" in the COD_DIAGNOSTICO column. While this value does appear in the real dataset, its dominance in the synthetic output suggests that the model failed to generalize well in this field, likely due to a combination of class imbalance and overfitting. This collapse significantly undermines the dataset's usefulness for diagnostic modeling and raises questions about the interpretability of otherwise strong metric results.

Moreover, although date fields appeared well formatted at first glance, a closer inspection is required to detect subtle but important logical errors. These include episodes that occur after a recorded date of death or medical exams conducted before the patient's birth, errors that cannot be detected by global distributional metrics alone. Metrics such as Correlation Similarity or PCA distance evaluate statistical alignment at the matrix level and are insensitive to per-patient inconsistencies or semantic violations.

In summary, ACTGAN produces structurally coherent and statistically faithful synthetic data that clearly outperforms DoppelGANger in terms of formatting and categorical diversity. Nevertheless, its inability to preserve logical consistency, temporal order and diagnostic variety highlights critical limitations. The model may be suitable for exploratory tasks, testing pipelines, or training models on less sensitive fields. However, its outputs should be used with caution in clinical applications that require strict temporal and semantic integrity.

Compared to DoppelGANger, ACTGAN achieved far greater diversity in high-cardinality fields such as analyses, medical acts and patient identifiers. While the synthetic dataset from DoppelGANger collapsed to only 72 unique patients and 2 distinct death dates across 30,000 records, ACTGAN generated was much more diverse and avoided such extreme mode collapse. In contrast, PARSynthesizer preserved structural fidelity by reproducing the original 300 patients and 239 death dates, but produced fewer records overall and

did not reach the same global diversity metrics as `ACTGAN`. These results illustrate the trade-offs: `ACTGAN` maximized statistical variety but lacked temporal coherence, while `DoppelGANger` suffered severe collapse and `PARSynthesizer` was more conservative yet structurally faithful.

Why Does ACTGAN Score So Well in Metrics?

The strong quantitative performance of `ACTGAN`, particularly in statistical fidelity metrics, can be attributed to several technical characteristics of its architecture and training procedure:

- **Tabular Attention Layers:** ACTGAN employs specialized attention mechanisms adapted to tabular data, enabling the model to capture complex interdependencies between features and improve column-level coherence.
- **Conditional Training:** The generator receives conditioning signals based on discrete attributes such as episode type or gender. This helps the model maintain realistic proportions and conditional relationships, boosting joint distribution integrity.
- **Class Imbalance Handling:** ACTGAN is designed to handle imbalanced categorical distributions more robustly than simpler GAN architectures, which contributes to better marginal integrity even in rare categories.
- **Adversarial Optimization:** The adversarial setup, with a discriminator penalizing unrealistic outputs, forces the generator to closely approximate the statistical properties of the real data, including means, variances, correlations and marginal distributions.

These architectural advantages explain why `ACTGAN` consistently outperforms other models like `ParSynthesizer` or `DoppelGANger` in integrity and utility scores. However, despite its strengths, `ACTGAN` does not enforce temporal consistency or logical constraints. Paradoxically, this lack of structural enforcement increases its statistical diversity by allowing the generation of combinations that, while potentially unrealistic, still boost performance in distributional metrics such as PCA distance, Wasserstein distance and mutual information.

5.4.1 More Methodologies: Experimental Results and Conclusions

Based on the weak results obtained in earlier experiments, we also explored alternative methodological setups in order to assess whether specific architectural changes or data restructuring could lead to improvements. In particular, we tested sequence-oriented models (e.g., HMM Synthesizer, EHR-M-GAN) and multi-table configurations (HMASynthesizer) to evaluate their capacity to capture temporal dynamics and relational consistency.

a) HMM Synthesizer

The HMM Synthesizer was applied to the sequentially structured version of the medical dataset, where each row represented a clinical event associated with a patient. The model successfully learned transition dynamics and emitted variables from latent states. However, several limitations were noted:

- During preprocessing, the model converts all categorical and binary variables into a one-hot format. This transformation generates one new column for each possible category of a variable. For example, the column `CONCELHO` with n distinct values (e.g., `CONCELHO_LISBOA`, `CONCELHO_SEIXAL`, etc.) is expanded into n separate binary columns. While this representation is mathematically convenient for HMMs, it results in a dramatic increase in dimensionality, leading to sparsity and higher memory consumption. In practice, this limits the number of patients and events that can be processed simultaneously and it reduces the scalability of the approach when applied to real-world EHR datasets with hundreds of categorical features.

- After training, the model generates values for these one-hot encoded columns in continuous format, often between 0 and 1 and occasionally above 1 due to the stochastic nature of sampling. Temporal variables such as dates are also emitted as raw numeric offsets (sometimes even negative), with no automatic reconversion to calendar format. As a result, the synthetic output is not directly interpretable and requires explicit post-processing to recover coherent categorical and temporal values.

- Evaluation metrics could not be applied directly due to output format mismatch.

- The HMM framework assumes that variables are conditionally independent given the latent state. In practice, this introduces implausible records and logical inconsistencies (e.g., `DTA_OBITO` before `DTA_NASC`, or simultaneous values such as `GENERO_F` = 0.5 and `GENERO_M` = 0.5).

Despite these limitations, the model is computationally efficient and interpretable. It may be useful for generating sequence-aware synthetic data if paired with a robust postprocessing pipeline.

Even after extensive postprocessing, the HMM Synthesizer yielded consistently low scores across key quality indicators. For instance, the *Overall Score* was only **5.98%**, while the *Synthetic Quality Score (SQS)* reached just **0.3402**. Additionally, the model failed completely in structural coherence, with a *Deep Structure Stability* of **0.0000**. These results confirm that traditional HMMs struggle to replicate the complexity and interdependence present in clinical datasets.

b) EHR-M-GAN

The EHR-M-GAN model was evaluated on the dataset in which each patient is observed across multiple time-ordered visits — to generate realistic multi-type trajectories. The model produced 9,000 synthetic *patient–visit* records, indexed by patient ID and visit number (`VISIT_IDX`).

Results:

- Output included original column formats (e.g., `GENERO`, `CONCELHO`) without requiring one-hot encoding;
- Continuous variables showed coherent values within realistic ranges;
- Dates were expressed as valid years and `DTA_OBITO` used `NaN` appropriately for living patients;
- Sequence order (`VISIT_IDX`) was preserved for each patient;
- No postprocessing was required to evaluate metrics.

Table 5.7: Evaluation Metrics Summary

Metric	Score
Column Shapes Score	12.12%
Column Pair Trends Score	3.84%
Overall Score	7.98%
Correlation Similarity	0.7171
Statistic Similarity	0.5839
Boundary Adherence	0.4000
Field Correlation Stability	0.5271
Field Distribution Stability	0.5536
Synthetic Quality Score (SQS)	0.3602
Attribute Inference Score	0.6322
PII Replay / Membership Inference Score	0.0000

Despite its architectural design for sequential data, EHR-M-GAN demonstrated limited effectiveness in reproducing the statistical properties of the clinical dataset. The evaluation metrics reveal low fidelity, weak correlation and distribution stability and a modest synthetic quality score. While the model maintains some level of temporal coherence and produces interpretable outputs, the overall performance suggests that EHR-M-GAN is not yet a reliable solution for practical synthetic data generation in healthcare contexts

c) HMASynthesizer - Multi-Table

To assess the quality of the multi-table synthetic data, we evaluated both `SDV` and `Gretel` models by analyzing key metrics across five structurally distinct tables. This per-table evaluation allows for a more granular understanding of each model's strengths and limitations with respect to different data complexities.

Table 5.8: Multi-Table Evaluation Results for SDV and Gretel

Table	SDV Score	Gretel SQS	Gretel Notes
`Patients`	82.17%	0.16	Strong privacy (PII = 1.0)
`Consultation_Diagnosis`	96.98%	0.15	No deep structure
`Consultations`	45.00%	0.40	Perfect PII protection
`Hospitalizations`	56.75%	0.44	Correlation = 0.99
`Emergencies`	26.00%	0.46	Strong structural scores

To further improve structural clarity and model performance, we experimented with dividing the dataset into nine distinct tables, closely aligned with the entities defined in the original ER diagram. This normalization step aimed to reduce sparsity, lower column dimensionality and better isolate relational logic.

While this restructuring did improve interpretability and maintain strong performance in structurally simple tables like `Patients` and `Consultation_Diagnosis`, the overall results remained largely consistent with previous runs. Tables with high complexity, such as `Hospitalizations` and `Emergencies`, continued to exhibit limitations in deep logic modeling - particularly under `Gretel`, where Deep Structure Stability consistently scored zero.

Nonetheless, the split facilitated clearer diagnostic insights by allowing per-entity evaluation and confirming that structural modeling benefits most from simplified, focused tables.

In summary, the multi-table approach with `HMASynthesizer` preserved referential integrity but did not translate into quality gains: (i) weak distributional fidelity and boundary adherence in episode tables, (ii) *Deep Structure Stability* near 0.0 and (iii) recurrent *PII replay*/inference flags that are sensitive to the configuration of *sensitive fields*. Normalizing to nine tables improved interpretability but not the core results. Overall, outcomes were weak and fell short of the best single-table runs, suggesting the need for additional conditioning/constraints and temporal post-processing.

6

Conclusion

The evaluation of the tested models revealed that different approaches offer distinct advantages depending on the defined objectives and the structural characteristics of the dataset. In particular, it was found that tabular models, such as `ACTGAN`, are effective in replicating statistical distributions and relationships between isolated attributes, showing overall positive results in this domain. However, the absence of a sequential component in these models severely limits their ability to coherently represent time-dependent events, an especially relevant factor in clinical contexts where multiple records are associated with the same patient.

On the other hand, sequential models such as `PARSynthesizer` and `DoppelGANger` proved to be conceptually more suitable for capturing temporal relationships between records. Nevertheless, the results obtained during the evaluation process exposed significant weaknesses in their ability to generalize the real structure of clinical data, particularly in maintaining logical consistency across episodes and in accurately representing multiple events associated with the same entity.

A deeper analysis of these limitations helps explain why none of the approaches fully met the requirements of temporal clinical datasets. `PARSynthesizer`, although straightforward to implement, treats each row as an independent record and thus fails to preserve sequences of events across patients. Its memory demands also make it unsuitable for larger datasets and it cannot enforce logical constraints such as preventing episodes after death. Some of these issues could potentially be mitigated by conditioning strategies (e.g., controlling for age or episode type), but the lack of native sequential modeling remains a critical obstacle.

`DoppelGANger` introduced explicit support for sequential data and showed promise in capturing temporal dependencies. However, it still produced inconsistent timelines, such as exams after death or events occurring out of logical order. These incoherencies arise from the model's inability to incorporate explicit constraints and its requirement that all sequences be of equal length, which necessitates padding or truncation.

`ACTGAN`, while successful in producing diverse and well-structured tabular attributes, shares the same structural limitation as ParSynthesizer: it cannot learn or preserve temporal

order. Moreover, certain columns suffered from severe mode collapse (e.g., over 90% of generated diagnosis codes defaulted to `NO_CODE`), substantially reducing the dataset's utility. Its execution exclusively via the Gretel Console further restricts parameter tuning and transparency, making it effectively a black-box solution with limited adaptability.

In sum, all three approaches demonstrated structural weaknesses that prevent them from generating clinically coherent synthetic datasets without additional intervention. Tabular models (`PARSynthesizer` and `ACTGAN`) lack temporal modeling altogether, while the sequential `DoppelGANger` cannot guarantee logical consistency across events. Furthermore, none of the models allow for the enforcement of clinical constraints, a critical feature when dealing with sensitive medical data.

Beyond the limitations of the generative models themselves, this work also highlighted shortcomings in the current evaluation mechanisms. While widely used metrics and validation frameworks provide useful insights at the level of marginal distributions and simple correlations, they remain insufficient to assess the clinical validity of synthetic datasets. Our contribution was to demonstrate, through a systematic evaluation, that existing proposals often fail precisely in the most complex scenarios, such as heterogeneous patient populations, one-to-many relationships and temporally constrained events.

Based on this analysis, it is concluded that the selection of the synthetic data generation model must be aligned with the specific objectives for which the data will be used. The absence of concrete use cases at this exploratory stage may have limited a better alignment between the preprocessing strategy, the model choice and the configuration of input parameters. Still, the results demonstrate that it is possible to successfully capture first order statistical and structural patterns from the original dataset, paving the way for future iterations more oriented toward specific goals and practical use cases.

We shared our results with both Gretel and SDV and received a response from SDV. According to SDV (personal communication, August 2025), the limitations, especially regarding constraint enforcement, arise from the current design of the Parsynthesizer algorithm and its transformation pipeline. A solution is envisioned but not yet prioritized and our feedback has been filed as a feature request.

Future Work

There are several avenues for future research that emerge from this study. One important direction is the development or adaptation of generative models that can explicitly enforce clinical rules during training and generation. For example, constraints such as ensuring that no episodes occur after a recorded death date or that diagnoses precede the corresponding treatments would improve the realism and reliability of synthetic patient records. A further direction concerns the exploration of more advanced architectures that may better capture the temporal and structural complexity of healthcare data.

Finally, this work also highlights the need for more robust evaluation frameworks. Current integrity, privacy and utility metrics, while informative, are limited in their ability

to capture clinical plausibility. Future efforts should therefore prioritize the creation of evaluation tools that do a better job at accurately assessing the quality of synthetic data. Such tools should move beyond simple statistical checks to provide a clearer picture of how well the generated datasets capture both the underlying structure and the clinical validity of the original data.

Bibliography

[1] J. M. Lourenço. *The NOVAthesis LaTeX Template User's Manual*. NOVA University Lisbon. 2021. URL: https://github.com/joaomlourenco/novathesis/raw/main/template.pdf (cit. on p. i).

[2] OECD. *Data-Driven Innovation: Big Data for Growth and Well-Being*. Accessed: 2025-02-11. Paris: OECD Publishing, 2015. URL: http://dx.doi.org/10.1787/9789264229358-en (cit. on p. 1).

[3] P. O. Coté, A. Nikanjam, N. Ahmed, D. Humeniuk, and F. Khomh. "Data Cleaning and Machine Learning: A Systematic Literature Review". In: *Automated Software Engineering* (2024). URL: https://link.springer.com/article/10.1007/s10515-024-00453-w (cit. on p. 1).

[4] U. Kartoun. "A Methodology to Generate Virtual Patient Repositories". In: *arXiv* (2016). URL: https://arxiv.org/abs/1608.00570 (cit. on p. 2).

[5] F. S. de Oliveira, C. R. Ferreira, and J. A. dos Santos Júnior. "Privacy in electronic health records: a systematic mapping study". In: *Journal of Public Health* 30.3 (2022). URL: https://link.springer.com/article/10.1007/s10389-022-01795-z (cit. on pp. 2, 8).

[6] E. Brown and M. Green. "Evaluation of Re-identification Risks in Data Anonymization Techniques Based on Population Uniqueness". In: *IEEE Transactions on Data Privacy* 10.3 (2020), pp. 123–134 (cit. on pp. 2, 8).

[7] Z. Whittaker. *Researchers Spotlight the Lie of 'Anonymous' Data*. https://techcrunch.com/2019/07/24/researchers-spotlight-the-lie-of-anonymous-data/. 2019 (cit. on p. 2).

[8] S. James, C. Harbron, J. Branson, and M. Sundler. In: *Journal of Biomedical Informatics* (2021). URL: https://link.springer.com/article/10.1007/s44163-021-00016-y (cit. on pp. 3, 10, 11).

[9] J. Walonoski, M. Kramer, J. Nichols, A. Quina, C. Moesel, D. Hall, C. Duffett, K. Dube, T. Gallagher, and S. McLachlan. "Synthea: An approach, method, and software mechanism for generating synthetic patients and the synthetic electronic health care record". In: *Journal of the American Medical Informatics Association* 25.3 (2018) (cit. on pp. 3, 15, 24).

[10] E. Choi, S. Biswal, B. Malin, J. Duke, W. Stewart, and J. Sun. "Generating Multi-Label Discrete Patient Records Using Generative Adversarial Networks". In: *Proceedings of the Machine Learning for Healthcare Conference*. PMLR, 2017. URL: https://proceedings.mlr.press/v68/choi17a.html (cit. on pp. 3, 13, 15–18).

[11] R. J. Hyndman and G. Athanasopoulos. *Forecasting: Principles and Practice*. OTexts, 2018. URL: https://otexts.com/fpp2/ (cit. on pp. 3, 15, 22).

[12] M. K. Baowaly, C. Lin, C. Liu, and K. Chen. "Synthesizing electronic health records using improved generative adversarial networks". In: *Journal of the American Medical Informatics Association* 26.3 (2019) (cit. on pp. 3, 18).

[13] L. Xu, M. Skoularidou, A. Cuesta-Infante, and K. Veeramachaneni. "Modeling Tabular Data Using Conditional GAN". In: *Advances in Neural Information Processing Systems (NeurIPS '19)*. Vol. 32. 2019. URL: https://proceedings.neurips.cc/paper/2019/file/254ed7d2de3b23ab10936522dd547b78-Paper.pdf (cit. on pp. 3, 13, 18, 19).

[14] Z. Lin, A. Jain, C. Wang, G. Fanti, and V. Sekar. "Using GANs for Sharing Networked Time Series Data: Challenges, Initial Promise, and Open Questions". In: (2020-10), 20 pages. DOI: 10.1145/3419394.3423643. URL: https://doi.org/10.1145/3419394.3423643 (cit. on pp. 3, 14).

[15] K. Zhang, K. Veeramachaneni, and N. Patki. "Sequential Models in the Synthetic Data Vault". In: *arXiv* (2022). URL: https://arxiv.org/abs/2207.14406 (cit. on pp. 3, 9, 11, 12, 15, 23).

[16] X. Luo. "A Method for Privacy-Safe Synthetic Health Data". In: *Academic Journal of Science and Technology* 10.1 (2020) (cit. on pp. 3, 8, 9, 16, 17, 19, 20).

[17] *VohcoLab: Voice Health and Communication*. Accessed: 2025-02-11. 2025. URL: https://vohcolab.org/ (cit. on p. 4).

[18] A. S. Mojsilović and J. L. Ferres. *Practical Synthetic Data Generation: Balancing Privacy and the Broad Availability of Data*. O'Reilly Media, 2022 (cit. on pp. 7, 10).

[19] D. Blumenthal and M. Tavenner. *The "Meaningful Use" Regulation for Electronic Health Records*. Massachusetts Medical Society, 2010. URL: http://www.ny3aap.org/documents/MeaningfulUseNEJM2010.pdf (cit. on p. 8).

[20] E. Parliament and C. of the European Union. *Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the Protection of Natural Persons with Regard to the Processing of Personal Data and on the Free Movement of Such Data (General Data Protection Regulation)*. 2016. URL: https://eur-lex.europa.eu/eli/reg/2016/679/oj (cit. on p. 8).

[21] A. Narayanan and V. Shmatikov. "Robust De-anonymization of Large Sparse Datasets". In: IEEE Computer Society, 2008. DOI: 10.1109/SP.2008.33 (cit. on p. 8).

[22] N. I. S. Organization. *Understanding Metadata*. Available at: https://www.niso.org/publications/understanding-metadata. Bethesda, MD: NISO Press, 2004 (cit. on p. 9).

[23] C. Dwork. "Differential Privacy". In: *International Colloquium on Automata, Languages, and Programming (ICALP)* (2006). DOI: 10.1007/11787006_1 (cit. on p. 9).

[24] J. R. Pleis. "Mixtures of Discrete and Continuous Variables: Considerations for Dimension Reduction". PhD thesis. University of Pittsburgh, 2018. URL: https://d-scholarship.pitt.edu/35090/1/Pleis_John_dissert_Aug2018.pdf (cit. on p. 10).

[25] E. D. P. Supervisor. "Synthetic Data". In: (2021). Accessed: 2025-02-11. URL: https://www.edps.europa.eu/press-publications/publications/techsonar/synthetic-data_en (cit. on p. 10).

[26] M. Corporation. *Synthea: Synthetic Patient Population Simulator*. Accessed: 2025-02-11. 2021. URL: https://synthetichealth.github.io/synthea/ (cit. on p. 11).

[27] C. Umesh, K. Schultz, M. Mahendra, S. Bej, and O. Wolkenhauer. "Preserving Logical and Functional Dependencies in Synthetic Tabular Data". In: *arXiv preprint arXiv:2409.17684* (2024) (cit. on p. 12).

[28] Y. Bengio, P. Simard, and P. Frasconi. "Learning Long-Term Dependencies with Gradient Descent is Difficult". In: *IEEE Transactions on Neural Networks*. 1994. URL: https://doi.org/10.1109/72.279181 (cit. on p. 14).

[29] H. L. S. International. *Health Level Seven International (HL7)*. Accessed: 2025-02-11. URL: https://www.hl7.org/ (cit. on p. 14).

[30] M. D. Wilkinson, M. Dumontier, I. J. Aalbersberg, et al. *The FAIR Guiding Principles for scientific data management and stewardship*. Accessed: 2025-02-11. Nature Publishing Group, 2016. URL: https://library.oapen.org/handle/20.500.12657/22918 (cit. on p. 14).

[31] V. S. Verykios, E. Bertino, I. N. Fovino, L. P. Provenza, Y. Saygin, and Y. Theodoridis. "A General Survey of Privacy-Preserving Data Mining Models and Algorithms". In: (2004). URL: https://www.researchgate.net/publication/226923272_A_General_Survey_of_Privacy-Preserving_Data_Mining_Models_and_Algorithms (cit. on pp. 14, 15).

[32] M. Hauskrecht. *Lecture Notes: Autoregressive Models for Time Series*. Accessed: 2025-02-11. URL: https://people.cs.pitt.edu/~milos/courses/cs3750/lectures/class16.pdf (cit. on pp. 15, 22).

[33] L. R. Rabiner. "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition". In: *Proceedings of the IEEE* 2 (), pp. 257–286. URL: https://ieeexplore.ieee.org/document/18626?arnumber=18626 (cit. on pp. 15, 25).

[34] DataCebo. *SDV Documentation – Multi Table Models*. Accessed: July 2025. 2024. URL: https://docs.sdv.dev/sdv/single-table-data/model-selection/multi-table (cit. on pp. 15, 26).

[35] J. Brickell and V. Shmatikov. "The Cost of Privacy: Destruction of Data-Mining Utility in Anonymized Data Publishing". In: *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. ACM, 2008. URL: https://doi.org/10.1145/1401890.1401904 (cit. on p. 16).

[36] S. Imtiaz, M. Arsalan, V. Vlassov, and R. Sadre. "Synthetic and Private Smart Health Care Data Generation using GANs". In: KTH Royal Institute of Technology. 2023 (cit. on p. 17).

[37] Z. Lin, A. Jain, C. Wang, G. Fanti, and V. Sekar. "Generating High-fidelity, Synthetic Time Series Datasets with DoppelGANger". In: *arXiv* (2019) (cit. on pp. 21, 22).

[38] A. W. Services. *O que são modelos autorregressivos?* Accessed: 2025-02-11. URL: https://aws.amazon.com/pt/what-is/autoregressive-models/ (cit. on p. 22).

[39] C. Lin, A. Jaech, X. Li, M. R. Gormley, and J. Eisner. "Limitations of Autoregressive Models and Their Alternatives". In: Association for Computational Linguistics, 2021. URL: https://aclanthology.org/2021.naacl-main.405/ (cit. on p. 22).

[40] J. Chen, D. Chun, M. Patel, E. Chiang, and J. James. "The validity of synthetic clinical data: a validation study of a leading synthetic data generator (Synthea) using clinical quality measures". In: *BMC Medical Informatics and Decision Making* 21.1 (2021) (cit. on p. 24).

[41] X. Jin and P. Xie. "EHR-M-GAN: Generating multi-label longitudinal electronic health records with missing values". In: *arXiv preprint arXiv:2106.10498* (2021). URL: https://arxiv.org/abs/2106.10498 (cit. on p. 25).

[42] Gretel.ai. *Synthetic Data Utility Report: Evaluate Synthetic Data*. Accessed: 2025-02-11. 2025. URL: https://docs.gretel.ai/ (cit. on pp. 28, 29).

[43] SDMetrics. *SDMetrics: Synthetic Data Evaluation Metrics*. Accessed: 2025-02-11. 2025. URL: https://docs.sdv.dev/sdmetrics (cit. on p. 28).

[44] S. Gonçalves, F. von Hafe, F. Martins, C. Menino, M. J. Guimarães, A. Mesquita, S. Sampaio, and A. R. Londral. "Case management intervention of high users of the emergency department of a Portuguese hospital: a before-after design analysis". In: *BMC Emergency Medicine* (2022) (cit. on p. 33).

[45] F. J. Anscombe. "Graphs in Statistical Analysis". In: *The American Statistician* (1973) (cit. on p. 66).

I

Data Analysis

This annex includes a set of complementary analyses performed on the clinical dataset. These exploratory insights, such as distribution patterns, temporal event overlaps and entity-level statistics, were conducted to deepen the understanding of the data prior to synthetic generation.

I.1 Overview of Patient and Episode Counts

The dataset includes the following numbers of patients and medical episodes:

- **Total number of unique patients:** 972
- **Total number of medical appointments:** 189,033
- **Total number of hospitalizations:** 5,036,938
- **Total number of emergency episodes:** 2,197,107

I.2 Diagnostic Records

To determine the total number of diagnoses, the following CSV files were filtered:

- *Consulta_Episodios_analises_exames_diagnosticos_medicamento*
- *Internamento_Episodios_analises_exames_gdh*

The results are as follows:

- **Total diagnoses in hospitalizations:** 5,036,938
- **Total diagnoses in medical appointments:** 486

I.3 Completeness of Attributes per Table

An analysis was conducted to assess the percentage of completed attributes in each CSV files.

- **Internamento_Medicamentos:** 100% of attributes are filled;
- **Internamento_Episodios_analises_exames_gdh:** 100% of attributes are filled;
- **Urgencias_Episodios_Triagem_analises_exames_medicamento:** 100% of attributes are filled.
- **Doentes_obito:** 100% of attributes are filled.

For the file *Consulta_Episodios_analises_exames_diagnosticos_medicamento*, several attributes have low completion rates (~**1.23%**), meaning that **98.77% of records for these fields are missing**.

I.3.1 Attributes with Low Completion Rate in the Medical Appointments Dataset

- CON_EPISODIO
- DTA_CONSULTA
- TIP_CONSULTA
- COD_MED_ESP
- ESPECIALIDADE
- COD_DIAGNOSTICO
- DES_DIAGNOSTICO
- TIPO_DIAGNOSTICO
- PRESCRICAO
- SEQ_PH_MED_PRES
- DATA_PRESC
- MEDICAMENTO
- NOME_MED
- TIPO_MD
- DOSE
- UNID_MED
- VIA_ADM
- FREQ_PRES
- HORARIO
- DT_INICIO
- DT_FIM
- OBS_MED

I.4 Exploratory Data Analysis and Patient Profiling

Figure I.1 shows the monthly distribution of the total number of *Análises* and *Exames* events across all patients. There is no particular month that stands out significantly in

terms of volume for either type of procedure. The number of *Análises* consistently exceeds the number of *Exames* throughout the year, a pattern that remains stable across all four years of data.

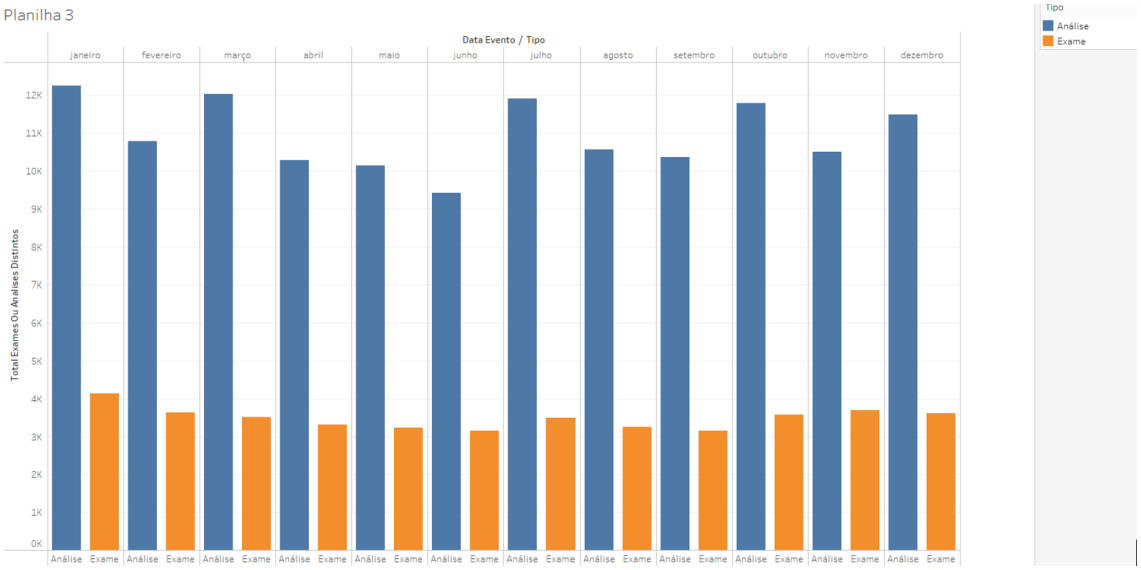


Figure I.1: Monthly distribution of the total number of laboratory tests (*Análises*) and medical exams (*Exames*) performed across all patients.

In terms of distinct records, however, the dataset contains a higher number of unique *Exames* (550) compared to unique *Análises* (288), as shown in Figure I.2.

123 total_exames_distintos	123 total_analises_distintos
550	288

Figure I.2: Total number of distinct laboratory tests (*Análises*) and medical exams (*Exames*) in the dataset.

Figure I.3 presents the gender distribution of the patient population. The dataset contains 442 female patients (48.04%) and 478 male patients (51.96%), indicating a balanced gender split. This equilibrium contributes to the dataset's representativeness across gender-related clinical patterns and reduces potential bias in model training.

Figure I.4 presents the average number of medical exams conducted per clinical episode, categorized by episode type. The results reveal that *Internamentos* (hospitalizations) have the highest average, with approximately 1.39 exams per episode. This is followed by *Urgência* (emergency episodes), with an average of 1.00 and lastly, *Consultas* (scheduled consultations), with only 0.67 exams per episode.

A-Z GENERO	123 count	123 percentage
F	442	48,0434782609
M	478	51,9565217391

Figure I.3: Gender distribution across the patient population.

	A-Z TIPO_EPISODIO	123 media_exames
1	Consulta	0,67
2	Internamento	1,39
3	Urgencia	1

Figure I.4: Average number of exams per episode type (*Consulta*, *Internamento*, *Urgência*).

Figure I.5 illustrates the average number of laboratory tests (*Análises*) performed per clinical episode, broken down by type. As expected, *Internamentos* (hospitalizations) exhibits the highest average with 1.04 tests per episode, followed by *Urgências* (emergency episodes) at 0.97 and *Consultas* (scheduled consultations) with an average of 0.83.

	A-Z TIPO_EPISODIO	123 media_analises
1	Internamento	1,04
2	Urgencia	0,97
3	Consulta	0,83

Figure I.5: Average number of laboratory tests (*Análises*) per episode type.

From the two previous figures, we can draw the following conclusions:

- **Consultations** tend to have the lowest number of associated diagnostic procedures, particularly in terms of medical exams.
- **Hospitalizations** are the episode type with the highest average number of both laboratory tests and medical exams.
- In **Emergency episodes**, there is a consistent pattern of performing approximately one exam and one analysis per episode, suggesting a standard diagnostic baseline applied in urgent care scenarios.

Figure I.6 compares the number of distinct clinical episodes and the total number of associated records for each episode type. Although `Consultas` (Consultations) account for the highest number of distinct episodes (16,832), they correspond to the lowest number of total rows (100,090). In contrast, `Internamentos` (Hospitalizations) represent the fewest distinct episodes (530), yet generate by far the most data - over 824,000 records - reflecting their higher complexity and the volume of associated events (e.g., multiple diagnostics, procedures, medications). `Urgência` (Emergencies) fall in between, with 9,220 episodes and 246,566 rows. These results highlight the varying data density and clinical richness associated with each type of episode.

	A-Z TIPO_EPISODIO	123 total_episodios
1	Consulta	16.832
2	Urgencia	9.220
3	Internamento	530

	A-Z TIPO_EPISODIO	123 total_linhas
1	Internamento	824.038
2	Urgencia	246.566
3	Consulta	100.090

Figure I.6: Distribution of clinical episode types by distinct episode count (left) and total number of associated rows (right).

Figure I.7 displays the annual distribution of clinical episodes between 2016 and 2020. The number of episodes increased from 2016 to 2017 and remained relatively stable through 2019. However, a noticeable decline is observed in 2020, where the total dropped to just 2,463 episodes.

A-Z ano	123 total_episodios
2016	3.243
2017	7.343
2018	6.877
2019	6.536
2020	2.463

Figure I.7: Total number of clinical episodes recorded per year (2016-2020).

Figure I.8 quantifies the number of patients who experienced all three types of clinical episodes: *Consultas*, *Urgência* and *Internamentos*. Out of a total of 920 patients analyzed, 187 had at least one episode of each type, representing approximately 20.33% of the population.

	123 pacientes_com_3_tipos	123 total	123 percentagem
1	187	920	20,33

Figure I.8: Patients with at least one episode of each type (*Consulta*, *Urgência* and *Internamento*).

Figure I.9 lists the most frequent diagnoses associated with patients prior to death, based on the unified diagnosis table. While the most prevalent entry is labeled as `UNKNOWN`, accounting for 312 occurrences, several specific conditions appear multiple times in deceased patient records. These include cardiac insufficiency (16), various forms of pneumonia (13) and chronic obstructive pulmonary disease (7). Other commonly observed diagnoses involve renal and respiratory complications, cardiovascular procedures and malignant neoplasms of the urinary tract.

	COD_DIAGNOSTICO	DES_DIAGNOSTICO	ocorrencias_em_doentes_que_faleceram
1	NO_CODE	UNKNOWN	312
2	194	Insuficiência cardíaca	16
3	139	Outras pneumonias	13
4	140	Doença pulmonar obstrutiva crónica	7
5	144	Sinais, sintomas e/ou diagnósticos minor respiratórios	7
6	461	Doenças malignas do rim e/ou vias urinárias	6
7	468	Outros diagnósticos, sinais e/ou sintomas no rim e/ou vias urinárias	6
8	201	Arritmia e/ou perturbações da condução cardíaca	5
9	446	Procedimentos uretrais e/ou transuretrais	5
10	197	Perturbações vasculares periféricas e/ou outras perturbações vasculares	4
11	254	Outros diagnósticos do aparelho digestivo	4
12	4151	EMBOLIA E ENFARTE PULMONAR	4
13	45	Acidente vascular cerebral e/ou oclusão pré-cerebral com enfarte	3
14	137	Infeções e/ou inflamações respiratórias major	3
15	174	Procedimentos cardiovasculares percutâneos com enfarte agudo do mio	3
16	175	Procedimentos cardiovasculares percutâneos sem enfarte agudo do mio	3
17	190	Enfarte agudo do miocárdio	3
18	228	Procedimentos para hérnia inguinal, femoral e/ou umbilical	3
19	280	Doenças alcoólicas do fígado	3

Figure I.9: Most common diagnoses associated with patients who passed away.

Figure I.10 shows the distribution of patients according to the number of distinct clinical episodes they are associated with. The majority of patients fall into two categories: those with between 10 and 25 episodes and those with between 25 and 50 episodes. A significant number also have fewer than 10 episodes. In contrast, relatively few patients have more than 100 associated episodes.

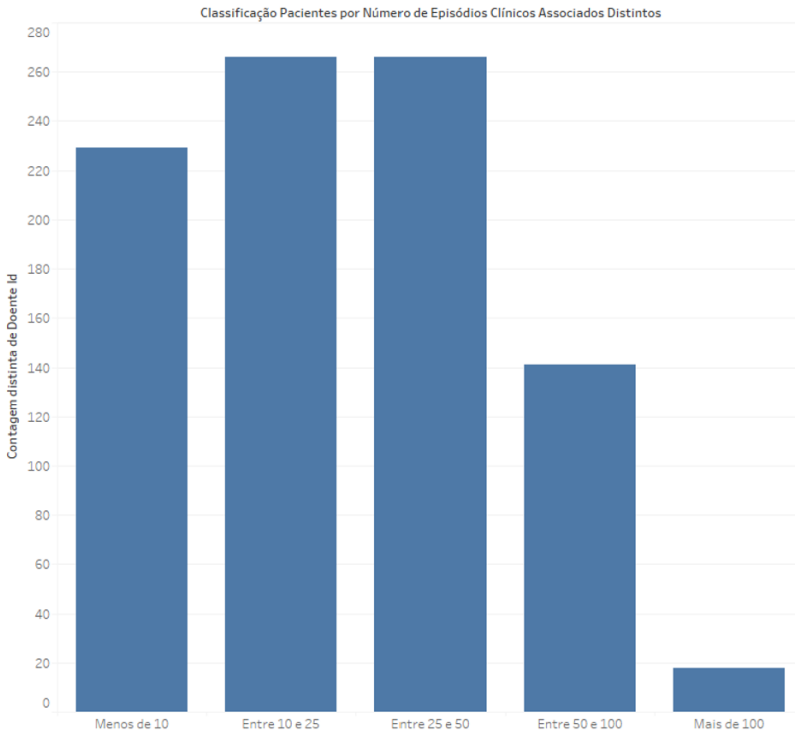


Figure I.10: Patient distribution by number of distinct associated clinical episodes (consultations, emergencies, hospitalizations).

These results suggest that patients do not contribute equally to the dataset in terms of episode volume. As such, when designing sampling strategies, for model training or synthetic data evaluation, it may not be appropriate to treat all patients as equally probable or representative. A uniform sampling approach could underrepresent high-frequency patients or overweight infrequent users.

Figure I.11 presents a simplified classification of patients based on the total number of distinct clinical episodes (consultations, emergencies, hospitalizations) into three usage tiers: **low users** (less than 20 episodes), **medium users** (between 20 and 50 episodes) and **extreme high users** (more than 50 episodes). The distribution shows that low and medium users are almost equally represented in the dataset, each accounting for nearly 400 patients. However, a smaller but still significant group - over 150 patients - falls into the extreme high user category.

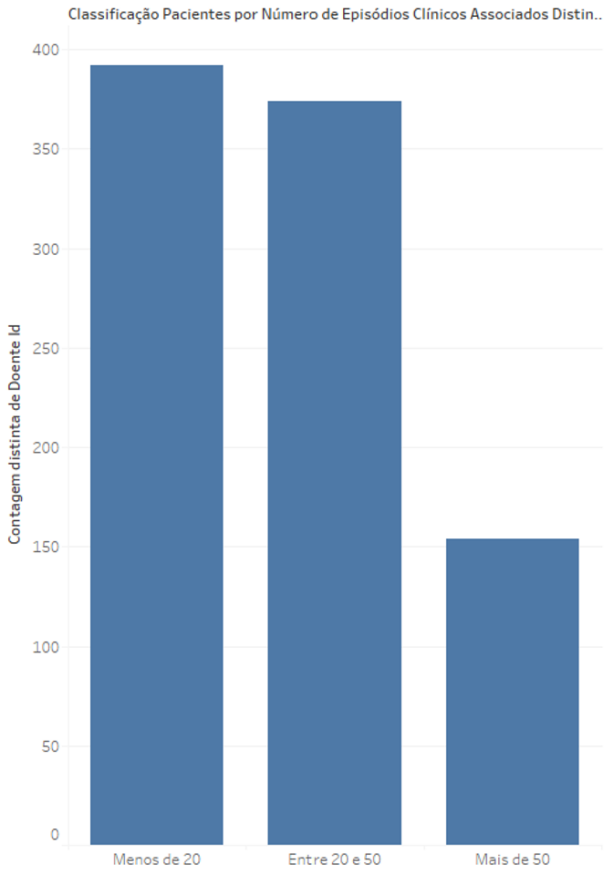


Figure I.11: Classification of patients into three groups based on number of distinct clinical episodes: low (<20), medium (20-50) and extreme high users (>50).

Figure I.12 illustrates the distribution of patient residences by municipality. The majority of patients are from the municipalities of Almada (579 patients) and Seixal (284 patients), which together account for over 80% of the entire patient population in the dataset. Other municipalities such as Sesimbra, Lisboa, Setúbal and Moita have marginal representation and most remaining regions appear with only one or two individuals.

	A-Z CONCELHO	123 num_pacientes
1	ALMADA	579
2	SEIXAL	284
3	SESIMBRA	11
4	LISBOA	7
5	SETUBAL	5
6	MOITA	4
7	AMADORA	3
8	PALMELA	2
9	OEIRAS	2
10	MAFRA	2
11	ABRANTES	2
12	VISEU	1
13	VILA NOVA FAMALICAO	1
14	VILA NOVA CERVEIRA	1
15	TORRES VEDRAS	1
16	SINTRA	1
17	SANTIAGO DO CACEM	1
18	SANTAREM	1
19	SALVATERRA DE MAGOS	1
20	PORTEL	1

Figure I.12: Patient residence distribution by municipality (*Concelho*).

Classification of Patients by Volume of Laboratory Tests and Exams

A breakdown of patient profiles based on the total number of distinct laboratory analyses and medical exams reveals important patterns in the intensity of clinical follow-up. These insights contribute to a more precise characterization of patient behavior and can inform sampling strategies, synthetic data modeling, or healthcare resource planning.

a) Number of Exams per Patient

The first analysis (Figure I.13) shows the number of exams associated with each patient. We observe that:

- The majority of patients underwent fewer than 10 distinct exams.
- A significant portion underwent between 10 and 25 exams.
- A smaller group underwent more than 25 exams, potentially reflecting more complex clinical cases or chronic conditions.

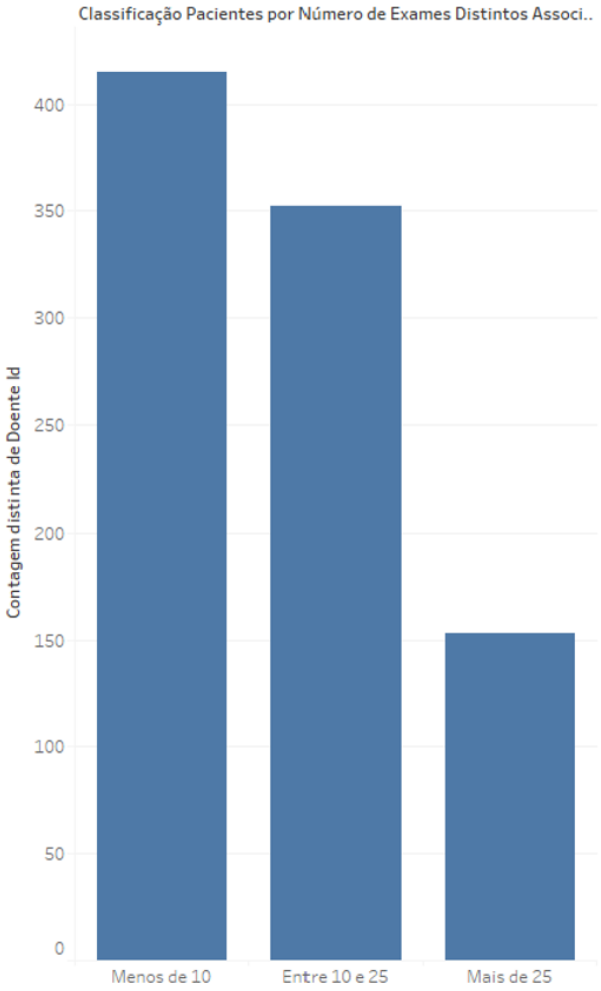


Figure I.13: Classification of Patients by Number of Distinct Exams.

b) Number of Analyses per Patient

Figure I.14 presents the distribution of the number of laboratory analyses per patient. Unlike the previous case:

- A large proportion of patients underwent more than 25 laboratory analyses.

- The remaining patients are almost equally distributed between the “less than 10” and “10 to 25” groups.

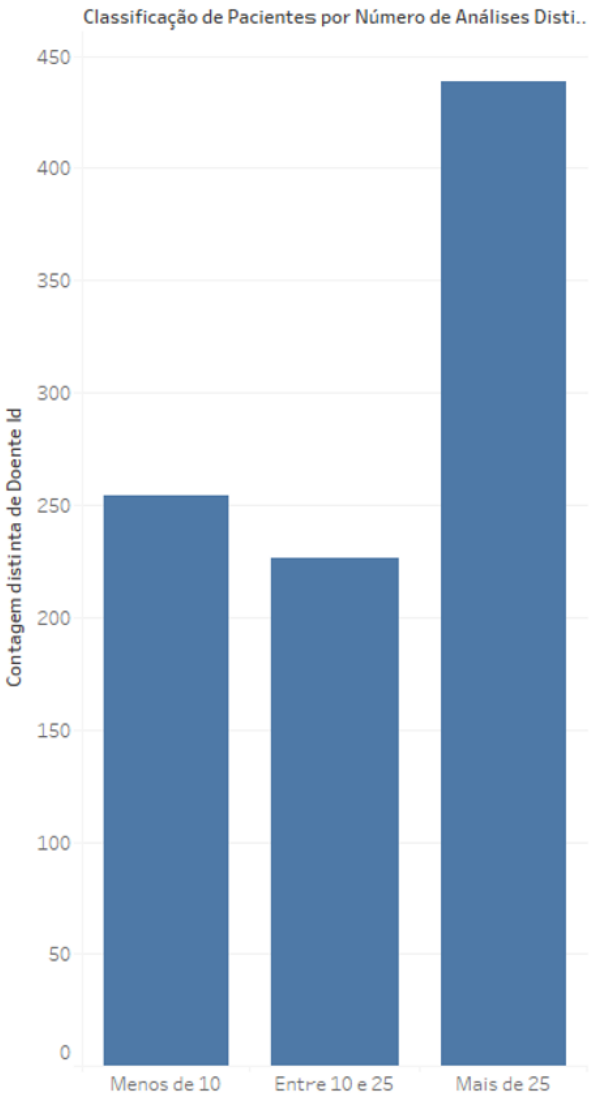


Figure I.14: Classification of Patients by Number of Distinct Laboratory Analyses.

c) Total Tests (Exams + Analyses) per Patient

The final analysis (Figure I.15) combines both categories - exams and analyses - to reflect the total number of distinct tests associated with each patient:

- Most patients fall within the 25 to 100 test range.
- A substantial group is also observed in the “less than 25” bracket.
- A smaller, yet relevant, segment underwent more than 100 distinct tests-likely corresponding to high-frequency or complex cases.

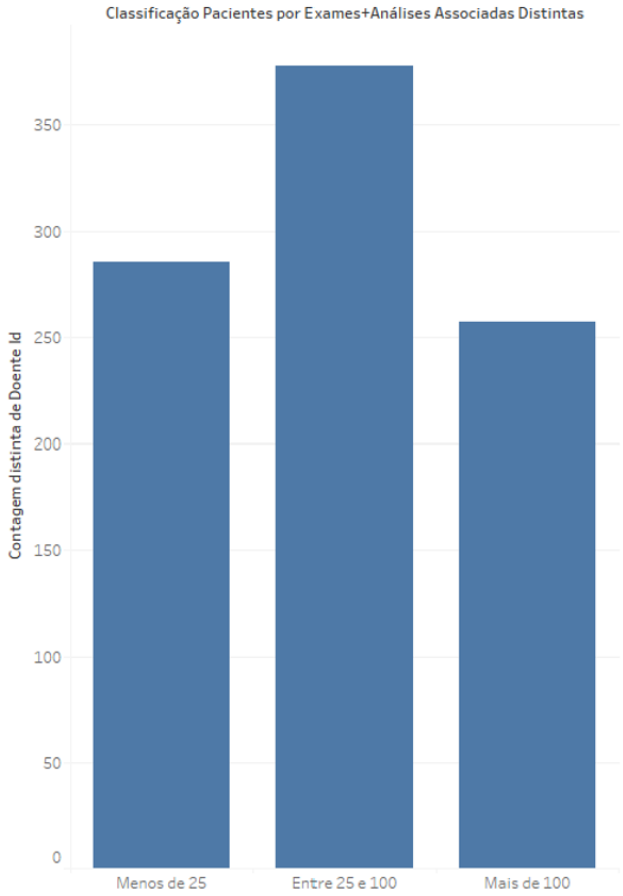


Figure I.15: Classification of Patients by Total Distinct Tests (Analyses + Exams).

From these analyses, we can conclude:

- Exams are less frequent than laboratory analyses across most patients.
- A substantial number of patients undergo a high volume of clinical testing, especially in terms of laboratory analyses.
- These findings support the classification of patients into usage tiers (e.g., low, medium, high), which can be valuable for sampling, synthetic generation strategies, or policy planning.

II

Annex II - Mapping of Entities and CVS Files

This annex provides a detailed description of the origin of the entities presented in the ER Model and the mapping between their English and Portuguese names, as they appear in the CSV files provided.

- The entity `Patient` - `Paciente` is derived from the CSV file `Doentes_obito` (1).
- The entity `Emergency` - `Urgência` is derived from the CSV file `Urgencias_Episodios_Triagem_analises_exames_medicamento` (2).
- The entity `Hospitalization` - `Internamento` is derived from the CSV file `Internamento_Episodios_analises_exames_gdh` (3).
- The entity `Medical Appointment` - `Consulta` is derived from the CSV file `Consulta_Episodios_analises_exames_diagnosticos_medicamento` (4).
- The entity `Medication` - `Medicamentos` is derived from the CSV file `Internamento_Medicamentos` (5).
- The entities `Exam` and `Laboratory Test` - `Exame` e `Análise` are derived from the CSV files (2, 3, 4).
- The entities `Diagnose` and `Medication` - `Diagnóstico` e `Medicação` are derived from the CSV files (3 and 4).
- Finally, the entity `Prescription` - `Prescrição` is derived from the CSV file (2).

III

Empirical Evaluation Script

```
import pandas as pd
import numpy as np
from scipy.stats import ks_2samp, chisquare
import unicodedata

# ---------------------------------------------------
#  funcoes para lidar com o problema das datas
#  ( ajuda no parsing/normalizacao y)
# ---------------------------------------------------

def strip_accents(text):
    if pd.isna(text):
        return text
    return ''.join(ch for ch in unicodedata.normalize('NFD', str(text)) if
    unicodedata.category(ch) != 'Mn')

def normalize_text_series(ser):
    ser = ser.astype(str).str.strip().str.lower().apply(strip_accents)
    ser = ser.str.replace(r"\s+", " ", regex=True)
    return ser

def to_datetime_pt(series):
    # 1) tenta DD/MM/YYYY
    s1 = pd.to_datetime(series, format="%d/%m/%Y", errors="coerce")
    # 2) fallback: dayfirst=True (apanha outros formatos PT), ou ISO
    s2 = pd.to_datetime(series, dayfirst=True, errors="coerce")
    s = s1.fillna(s2)
```

```
    return s



# ---------------------------------------------------
# Step 1: Load Datasets (real and syntheticc )
# ---------------------------------------------------

def load_datasets(real_path, synth_path):
    real_df = pd.read_csv(real_path, sep=";", engine="python")
    synth_df = pd.read_csv(synth_path, sep=";", engine="python")

    #checkar as colunas de ambos os datasets
    print("Real dataset columns:", list(real_df.columns))
    print("Synthetic dataset columns:", list(synth_df.columns))
    return real_df, synth_df



# ---------------------------------------------------
# Step 2: Date Restriction Checker
# ---------------------------------------------------

# recebe um df, 2 colunas e um operador de comparacao
def compare_dates(df, col1, col2, comparison=">="):
    result = {"restriction": f"{col1} {comparison} {col2}"}
    if col1 not in df.columns or col2 not in df.columns:
        result["error"] = f"Column(s) not found: {col1}, {col2}"
        return result

    # trabalhar numa copia, converter usando formato PT e ignorar
    nAs nas comparacoes
    tmp = df[[col1, col2]].copy()
    tmp[col1] = to_datetime_pt(tmp[col1])
    tmp[col2] = to_datetime_pt(tmp[col2])
    mask = tmp[col1].notna() & tmp[col2].notna()
    if mask.sum() == 0:
        result.update({
            "total_rows": len(df),
            "rows_compared": 0,
```

```
            "ignored_rows_na": len(df),
            "valid_rows": 0,
            "invalid_rows": 0,
            "valid_percentage": 0.0
        })
        return result

    a, b = tmp.loc[mask, col1], tmp.loc[mask, col2]
    operators = {
        ">":  a >  b,
        ">=": a >= b,
        "<":  a <  b,
        "<=": a <= b,
        "==": a == b,
    }
    if comparison not in operators:
        result["error"] = f"Unsupported operator: {comparison}"
        return result

    valid = operators[comparison]

    # aplica o operador escolhido;  dicionario
    valid_rows = int(valid.sum())
    total_compared = int(mask.sum())
    invalid_rows = total_compared - valid_rows
    result.update({
        "total_rows": len(df),
        "rows_compared": total_compared,
        "ignored_rows_na": int((~mask).sum()),
        "valid_rows": valid_rows,
        "invalid_rows": invalid_rows,
        "valid_percentage": round((valid_rows / total_compared) * 100, 2)
    })
    return result


# ------------------------------------------
# Kolmogorov-Smirnov Test - Explicacao(net)
# ------------------------------------------
# This function uses the Kolmogorov-Smirnov (KS) test to
```

```
# compare the distributions
# of a numerical column between the real and synthetic datasets.
#
# The KS test answers the question: "Do these two samples come
# from the same distribution?"
#
# It returns:
#   - KS statistic (D): the maximum difference between the cumulative
#     distributions.
#       - Values close to 0 -> distributions are very similar.
#       - Values closer to 1 -> distributions are very different.
#
#   - p-value: probability that the difference is due to chance.
#       - p > 0.05 ->  we fail to reject H0 -> Distributions are
#       considered similar.
#       - p <= 0.05 -> we reject H0 -> Distributions are
#       significantly different.
#
# In this context, we use it to validate whether the synthetic data
# preserves the statistical characteristics of the real dataset.



# ------------------------------------------
# Step 3: Distribution Comparison - Colunas Numericas
# ------------------------------------------

def compare_column_distributions(real_df, synth_df, columns):
    results = {}
    for col in columns:
        #garantir que a coluna existe em ambos os datasets
        if col not in real_df.columns or col not in synth_df.columns:
            results[col] = {"error": "Column not found"}
            continue
        real_data = pd.to_numeric(real_df[col], errors="coerce").dropna()
        synth_data = pd.to_numeric(synth_df[col], errors="coerce").dropna()
        try:
            #aplicar Kolomogorov-Smirnov
            if len(real_data) == 0 or len(synth_data) == 0:
                results[col] = {
```

```
                    "error": "No comparable numeric data (non-NA) for KS",
                    "n_real": int(len(real_data)),
                    "n_synth": int(len(synth_data))
                }
                continue
            stat, p_value = ks_2samp(real_data, synth_data)
            results[col] = {
                "ks_statistic": round(stat, 4),
                "p_value": round(p_value, 5),
                "distribution_match": p_value > 0.05,
                "n_real": int(len(real_data)),
                "n_synth": int(len(synth_data))
            }
        except Exception as e:
            results[col] = {"error": str(e)}
    return results
# Retorna estatistica KS, p-value e um boolean distribution_match (p>0.05).


# ──────────────────────────────────────────────
# Step 4: Distribution Comparison - Categoricas
# ──────────────────────────────────────────────

def compare_categorical_distributions(real_df, synth_df, column, top_n=10):
    if column not in real_df.columns or column not in synth_df.columns:
        return {"error": "Column not found"}

    # calcular frequencias relativas
    real_counts = real_df[column].astype(str).value_counts()
    synth_counts = synth_df[column].astype(str).value_counts()

    # Top-N do real e bucket "__OTHER__" para fechar as somas
    top_idx = real_counts.head(top_n).index.tolist()

    def split_counts(counts, top):
        top_part = counts.reindex(top, fill_value=0)
        other = counts[~counts.index.isin(top)].sum()
        top_part["__OTHER__"] = other
        return top_part
```

```
    real_split = split_counts(real_counts, top_idx)
    synth_split = split_counts(synth_counts, top_idx)

    f_exp = real_split.values.astype(float)
    f_obs = synth_split.reindex(real_split.index).values.astype(float)

    try:
        stat, p = chisquare(f_obs=f_obs, f_exp=f_exp)
        return {
            "real_top_values": (real_split / real_split.sum()).to_dict(),
            "synth_top_values": (synth_split / synth_split.sum()).to_dict(),
            "chi2_statistic": round(stat, 4),
            "p_value": round(p, 5),
            "distribution_match": p > 0.05
        }
    except Exception as e:
        return {"error": str(e)}


# ---------------------------------------------
# Step 5: Datas -> Derived Attributes
# ---------------------------------------------

def add_derived_columns(df):
    df["DTA_NASCIMENTO"] = to_datetime_pt(df.get("DTA_NASCIMENTO"))
    df["DTA_EPISODIO"] = to_datetime_pt(df.get("DTA_EPISODIO"))
    df["DTA_ALTA_ADMIN"] = to_datetime_pt(df.get("DTA_ALTA_ADMIN"))
    df["IDADE"] = ((df["DTA_EPISODIO"] -
    df["DTA_NASCIMENTO"]).dt.days // 365).astype("Int64")
    df["DURACAO_EPI"] = (df["DTA_ALTA_ADMIN"] -
    df["DTA_EPISODIO"]).dt.days.astype("Int64")
    return df



# ---------------------------------------------
# Step 6: Static Field Consistency
# ---------------------------------------------

def check_static_field_consistency(df, group_col, static_fields):
```

```
    violations = {}
    grouped = df.groupby(group_col)
    for field in static_fields:
        inconsistent_ids = grouped[field].nunique()[lambda x:
        x > 1].index.tolist()
        violations[field] = {
            "total_inconsistent_patients": len(inconsistent_ids),
            "inconsistent_ids_sample": inconsistent_ids[:5]
        }
    return violations

# ---------------------------------------------------
# Step 7: Death Date Anomaly Check
# ---------------------------------------------------

def analyze_death_distribution(df, column="DTA_OBITO"):
    if column not in df.columns:
        return {"error": f"{column} not found"}
    dates = to_datetime_pt(df[column]).dropna()
    repeated = dates.value_counts()[lambda x: x > 1]
    return {
        "total_death_records": len(dates),
        "unique_dates": dates.nunique(),
        "top_repeated_dates": repeated.head(5).to_dict()
    }




#---------------------------------------------------
#---------------------------------------------------
# ---------------------------------------------------
# Main Pipeline
# ---------------------------------------------------

def main():
    real_path = "real_dataset.csv"  #colocar o dataset real que queremos
    como input
    synth_path = "synthetic_dataset.csv" #colocar o dataset
    sintetico que
```

```
queremos avaliar


real_df, synth_df = load_datasets(real_path, synth_path)

real_df = add_derived_columns(real_df)
synth_df = add_derived_columns(synth_df)

print("\n Date restrictions....")
date_checks = [
    ("DTA_EPISODIO", "DTA_ALTA_ADMIN", "<="),
    ("DTA_EPISODIO", "DTA_OBITO", "<="),
    ("DTA_EPISODIO", "DTA_ANALISE", "<="),
    ("DTA_EPISODIO", "DTA_EXAME", "<="),
    #  birth date before death date (apenas quando existe obito)
    ("DTA_NASCIMENTO", "DTA_OBITO", "<="),
]


for col1, col2, op in date_checks:
    result = compare_dates(synth_df, col1, col2, op)
    if "error" in result:
        print(f" - {result['restriction']}:  ERROR -
        {result['error']}")
    else:
        print(f" - {result['restriction']}:
        {result['valid_percentage']}% valid
        ({result['invalid_rows']} violations) "
              f"[compared={result.get('rows_compared', 'NA')},
              ignored_na={result.get('ignored_rows_na','NA')}]")



print("\n  Numerical distributions (KS test)...")
numeric_cols = ["IDADE", "DURACAO_EPI"]
dist_results = compare_column_distributions(real_df,
synth_df, numeric_cols)
for col, res in dist_results.items():
    if "error" in res:
        print(f" - {col}:  ERROR - {res['error']}
```

```
            (n_real={res.get('n_real','?')},
            n_synth={res.get('n_synth','?')})")
        else:
            match = " Match" if res["distribution_match"] else "Mismatch"
            print(f" - {col}: KS={res['ks_statistic']} |
            p={res['p_value']} --> {match} "
                  f"(n_real={res.get('n_real')},
                  n_synth={res.get('n_synth')})")



print("\n Categorical distribution (diagnosis codes)...")
cat_result = compare_categorical_distributions(real_df, synth_df,
column="COD_DIAGNOSTICO")
if "error" in cat_result:
    print(f" - COD_DIAGNOSTICO:  ERROR - {cat_result['error']}")
else:
    match = " Match" if cat_result["distribution_match"]
    else "Mismatch"
    print(f" - COD_DIAGNOSTICO: \u03C7\u00B2=
    {cat_result['chi2_statistic']:.4f} |
    p={cat_result['p_value']:.5g} --> {match}")



print("\n Static field consistency per patient...")
static_fields = ["DTA_NASCIMENTO", "DTA_OBITO", "GENERO", "CONCELHO"]
consistency_results = check_static_field_consistency(synth_df,
"DOENTE_ID", static_fields)
for field, res in consistency_results.items():
    print(f" - {field}: {res['total_inconsistent_patients']}
    patients inconsistent")

print("\n  Death date distribution...")
death_stats = analyze_death_distribution(synth_df)
if "error" in death_stats:
    print(f" - ERROR - {death_stats['error']}")
else:
    print(f" - Unique DTA_OBITO dates: {death_stats['unique_dates']}
```

```
        (Repeated: {len(death_stats['top_repeated_dates'])})")


# ---------------------------------------------------------------
# Run it
# ---------------------------------------------------------------

if __name__ == "__main__":
    main()
```