UNIVERSITÀ DEGLI STUDI DI TRIESTE

XXXIV CICLO DEL DOTTORATO DI RICERCA IN

INGEGNERIA INDUSTRIALE E DELL'INFORMAZIONE

Generating Privacy-Compliant, Utility-Preserving Synthetic Tabular and Relational Datasets Through Deep Learning

Settore scientifico-disciplinare : **ING-INF/05 SISTEMI DI ELABORAZIONE DELLE INFORMAZIONI**

DOTTORANDO

Daniele Panfilo

COORDINATORE

Prof. Alberto Tessarolo

SUPERVISORE DI TESI

Prof. Eric Medvet

ANNO ACCADEMICO 2020/2021

Acknowledgement

"Fake it 'till you make it"
Aphorism of unknown origin

When I started this research, I could not have imagined it would take me to where I am today: in my office at my own company *Aindo*, where our mission is to help the world embrace the power of artificial intelligence. None of this would have been possible without the work I conducted at the University of Trieste. I would like to take this opportunity to thank all those whose contributions have been invaluable.

First and foremost, I would like to thank my supervisor, Professor Eric Medvet. Throughout the PhD, Eric has been nothing but patient and willing to share his expertise. His devotion to science and creativity has continuously inspired me, and I hope that some of it is reflected in my own work.

I would also like to thank my co-founders at Aindo, Sebastiano, Pietro, and Alexander. When we first embarked on this bold journey, we could not have dreamed of things unraveling as they have. While the company expanded, we have maintained our enthusiasm and belief in artificial intelligence. I cannot wait to see where we venture off to next!

I would also like to thank those who have joined Aindo for sharing our dream and for their excellent work: Alessandro, Andrea, Beste, Borut, Carlo, Ciro Antonio, Fabio, Gabriele, other Gabriele, Gianluca, Johannes Jasper, Matteo, Simone, Stefano, Vanessa. I am indebted to Misal and Roberto from Vertis S.G.R. for their belief in Aindo.

I want to thank Francesca. You continuously provide me with love, support, trust, optimism and tea and without you, none of this would have been possible. I thank my parents for being there for me whenever I need them. Last but not least, I want to thank my friends: Anowar, Edoardo, Elisa, Filippo B., Filippo T.,Gabriela, Giulia, Irene, Marco, Roberta.

Summary

Two trends have rapidly been redefining the artificial intelligence (AI) landscape over the past several decades. The first of these is the rapid technological developments that make increasingly sophisticated AI feasible. From a hardware point of view, this includes increased computational power and efficient data storage. From a conceptual and algorithmic viewpoint, fields such as machine learning have undergone a surge and synergies between AI and other disciplines have resulted in considerable developments.

The second trend is the growing societal awareness around AI. While institutions are becoming increasingly aware that they have to adopt AI technology to stay competitive, issues such as data privacy and explainability have become part of public discourse. Combined, these developments result in a conundrum: AI can improve all aspects of our lives, from healthcare to environmental policy to business opportunities, but invoking it requires the use of sensitive data.

Unfortunately, traditional anonymization techniques do not provide a reliable solution to this conundrum. They are insufficient in protecting personal data, but also reduce the analytic value of data through distortion. However, the emerging study of *deep-learning generative models* (DLGM) may form a more refined alternative to traditional anonymization. Originally conceived for image processing, these models capture probability distributions underlying datasets. Such distributions can subsequently be sampled, giving new data points not present in the original dataset. However, the overall distribution of *synthetic data*sets, consisting of data sampled in this manner, is equivalent to that of the original dataset.

In this thesis, we study the use of DLGM as an enabling technology for wider AI adoption. To do so, we first study legislation around data privacy with an emphasis on the European Union. In doing so, we also provide an outline of traditional data anonymization technology. We then provide an introduction to AI and deep-learning. Two case studies are discussed to illustrate the field's merits, namely image segmentation and cancer diagnosis. We then introduce DLGM, with an emphasis on variational autoencoders. The application of such methods to tabular and relational data is novel and involves innovative preprocessing techniques. Finally, we assess the developed methodology in reproducible experiments, evaluating both the analytic utility and the degree of privacy protection through statistical metrics.

Sommario

Due tendenze hanno rapidamente ridefinito il panorama dell'intelligenza artificiale (IA) negli ultimi decenni. La prima è il rapido sviluppo tecnologico che rende possibile un'intelligenza artificiale sempre più sofisticata. Dal punto di vista dell'hardware, ciò include una maggiore potenza di calcolo ed una sempre crescente efficienza di archiviazione dei dati. Da un punto di vista concettuale e algoritmico, campi come l'apprendimento automatico hanno subito un'impennata e le sinergie tra l'IA e le altre discipline hanno portato a sviluppi considerevoli.

La seconda tendenza è la crescente consapevolezza della società nei confronti dell'IA. Mentre le istituzioni sono sempre più consapevoli di dover adottare la tecnologia dell'IA per rimanere competitive, questioni come la privacy dei dati e la possibilità di spiegare il funzionamento dei modelli di apprendimento automatico sono diventate parte del dibattito pubblico. L'insieme di questi sviluppi genera però una sfida: l'IA può migliorare tutti gli aspetti della nostra vita, dall'assistenza sanitaria alla politica ambientale, fino alle opportunità commerciali, ma poterla sfruttare adeguatamente richiede l'uso di dati sensibili.

Purtroppo, le tecniche di anonimizzazione tradizionali non forniscono una soluzione affidabile a suddetta sfida. Non solo non sono sufficienti a proteggere i dati personali, ma ne riducono anche il valore analitico a causa delle inevitabili distorsioni apportate ai dati. Tuttavia, lo studio emergente dei modelli generativi ad apprendimento profondo (MGAP) può costituire un'alternativa più raffinata all'anonimizzazione tradizionale. Originariamente concepiti per l'elaborazione delle immagini, questi modelli catturano le distribuzioni di probabilità sottostanti agli insiemi di dati. Tali distribuzioni possono essere successivamente campionate, fornendo nuovi campioni di dati, non presenti nel set di dati originale. Tuttavia, la distribuzione complessiva degli insiemi di dati sintetici, costituiti da dati campionati in questo modo, è equivalente a quella del set dei dati originali.

In questa tesi, verrà analizzato l'uso dei MGAP come tecnologia abilitante per una più ampia adozione dell'IA. A tal scopo, verrà ripercorsa prima di tutto la legislazione sulla privacy dei dati, con particolare attenzione a quella relativa all'Unione Europea. Nel farlo, forniremo anche una panoramica delle tecnologie tradizionali di anonimizzazione dei dati. Successivamente, verrà fornita un'introduzione all'IA e al deep-learning. Per illustrare i meriti di questo campo, vengono discussi due casi di studio: uno relativo alla segmentazione delle

immagini ed uno relativo alla diagnosi del cancro. Si introducono poi i MGAP, con particolare attenzione agli autoencoder variazionali. L'applicazione di questi metodi ai dati tabellari e relazionali costituisce una utile innovazione in questo campo che comporta l'introduzione di tecniche innovative di pre-elaborazione. Infine, verrà valutata la metodologia sviluppata attraverso esperimenti riproducibili, considerando sia l'utilità analitica che il grado di protezione della privacy attraverso metriche statistiche.

Contents

Chapter 1

Introduction

1.1 Motivation

An essential part of the European Commission (EC) digital strategy is titled "Shaping Europe's Digital Future" [1]. It aims at setting the worldwide standard for horizontal and vertical data sharing through the "Towards Common Data Spaces" initiative [2]. The main goal is to facilitate an efficient flow of data within the EU for public as well as private sector data. However, establishing these common data spaces is difficult to combine with the EC goal of giving citizens more control and protection of their data.

The absence of such common data spaces restricts business and research opportunities, as use of data is a determining factor in the productivity of organizations. Advances in data management are typically easily incorporated into a practical setting. Data analysis, however, tends to be a more convoluted process. While Machine Learning (ML) tools provide sophisticated analytic opportunities, for most small and medium-sized enterprises they are perceived as esoteric, requiring a very specific skill set to be put to good use. In consequence, data analysis is often outsourced to a consultancy company or a dedicated internal department. However, topics such as data protection and confidentiality have entered public discourse and raised concern. This has forced organizations to reflect on the role of data in their operations. Legislation such as the General Data Protection Regulation (GDPR) and the California Consumer Privacy Act (CCPA) has further complicated the use of data, particularly its transfer to third-party analysts.

Anonymization techniques are a common way of coping with these risks. Anonymization consists in masking, permuting, generalizing and/or distorting records to make their origins opaque. Unfortunately, anonymization techniques have been shown to be be susceptible to three seminal privacy risks (re-identification, linkage and parameter inference attacks) [3]. Moreover, the degree of anonymization directly affects the quality of the information still contained in the dataset. More recent methods based on differential privacy offer

more protection, but limit the number of use cases, excluding machine learning and artificial intelligence.

Deep learning (DL)-based *generative models* provide a reliable alternative to data anonymization. Generative models infer patterns from a real dataset, that can subsequently be used to generate highly realistic, yet fully artificial data points. Resulting artificial datasets closely mimic the real datasets, but do not contain any sensitive information. As such, artificial datasets can potentially be transferred without breaching privacy legislation, while the quality of the extracted information is on a par with the original dataset.

To date, most studies on generative models were conducted in the context of image generation (e.g., [4, 5]). These studies have introduced rigorous methodological frameworks and pivotal insights. However, the focus on image generation is restrictive, as most business cases are based on data in tabular form. That is, business data is organized in one or more tables in which rows represent entities, individuals, or individual actions and values on columns represent their properties. When the data span across more than one table, there are usually dedicated columns for linking entities from one table to another and multiple rows and columns can have intricate inter-dependencies. This case is usually referred to as *relational data*. A further difficulty in generating tabular data is the presence of several data types [6] (for example, categorical and continuous attributes) and the fact that some cells (column values for rows) can be empty, i.e., there can be *missing values*. These peculiarities call for a novel approach.

In this thesis, we introduce deep-learning generative models for tabular and relational data. To do so, we first illustrate the demand for reliable privacy enhancement technology, suitable for machine learning and artificial intelligence applications. We then outline the current legal landscape surrounding data privacy, with an emphasis on the European Union. Combined, these topics provide clear evidence that there is demand for innovative privacy technologies. We then design and describe in detail a synthetic data generation pipeline for the case when original data is tabular (possibly relational) and exhibits the practically relevant peculiarities described above: heterogeneous data types and missing values. We also propose measuring its outputs along two axes: (a) the degree to which the technique is able to generate data that is useful as a replacement for the original data while working with ML services (*utility-preservation*) and (b) the degree to which the technique prevents the disclosure of information about the entities represented in the original data from the synthetic data (*disclosure-averseness*). Our methods are evaluated in reproducible experiments with publicly available and randomly generated datasets. Results indicate that our approach is able to generate synthetic data with strong utility-preservation and disclosure-averseness.

1.2 Research Questions

The objective of this thesis is the establishment of effective methods for generating synthetic tabular and relational data. Such data should be reliable both

in its *utility preservation* and in its *disclosure averseness*. We also aim to show that there is an express need for the resulting technology in practice. A multi-disciplinary approach is taken to disclosure averseness, as this property is not merely a mathematical concern, but perhaps more so a legal one. Combined, these considerations lead us to the following research questions.

1.2.1 Research Question 1

What is machine learning and does the advent of its methods result in a need for reliable privacy enhancing technologies?

To answer this question, we provide a bottom-up background of machine learning, from the basic definitions to a high-level description of deep-learning methods. This background is illustrated through two practical use-cases: an oncological case and an image segmentation case. Combined, these consideration provide a conceptual understanding of methods developed in later chapters, but also showcase the dire need of secure data mobility across sectors and industries.

1.2.2 Research Question 2

Under what conditions is a dataset considered legally void of sensitive information? which approaches exist to achieve these conditions? and do these approaches meet the conditions?

To answer this question, we study the existing legislation on data privacy, with an emphasis on the European Union (EU). By introducing the General Data Protection Regulation (GDPR), the EU was one of the first major players in data privacy law. The GDPR is often used as a blueprint for privacy legislation elsewhere. Besides legislation, we also study the occurrence of data breaches in practice. An overview of the existing data privacy enhancement techniques is then provided, along with remarks on their merits and shortcomings.

1.2.3 Research Question 3

Can deep-learning models be applied for successfully generation of corporate data? if so, how can their effectiveness (utility and privacy) be measured?

To answer this question, we study existing frameworks for synthetic data generation. We identify variational autoencoders (VAE) as the most promising one for corporate datasets, that are typically tabular or relational in nature. We study advanced VAE models to facilitate optimal performance in these use cases. This includes a novel adaptation of *introspective VAE* for relational data. Preprocessing techniques for various data types are also discussed. Metrics to evaluate the degree to which utility and privacy are preserved are introduced.

1.2.4 Research Question 4

How effective are variational autoencoder models for synthetic data generation in practice?

To answer this question, we implement our models and evaluate them experimentally. To do so, we use three publically aailable datasets for tabular data. For relational data, we use randomly generated data. After applying the deep-learning generative models, we evaluate the performance metrics for utility preservation and disclosure averseness. This provides insight into the effectiveness of the models.

1.3 Structure of the Thesis

The remainder of this thesis is structured as follows: In Chapter 2, we answer Research Question 1, outlining the field of machine learning and providing practical motivation for privacy enhancing technologies. In Chapter 3, we answer Research Question 2, providing an overview of data privacy legislation, how prolific data breaches are, and what technologies exist for data privacy protection. In Chapter 4, we derive the deep-learning methods for synthetic data generation and introduce metrics to quantify the performance of their outputs. In Chapter 5, the effectiveness of the methods is evaluated experimentally. The thesis is concluded with a discussion of the obtained results and suggestions for future research in Chapter 6.

This structure, along with precedence relations between chapters, is visualized in Figure 1.1.

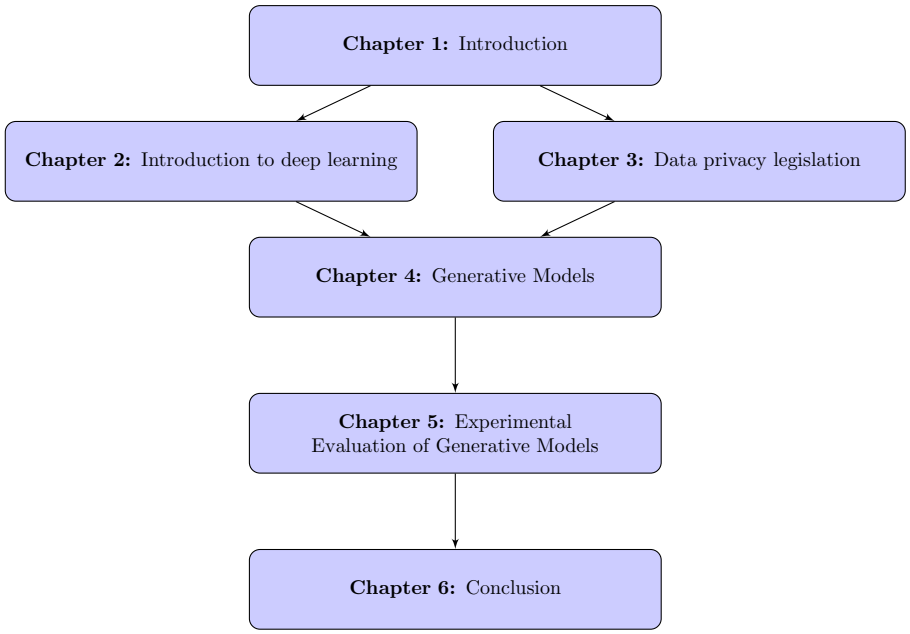


Figure 1.1: Structure of the thesis, with precedence relations between chapters.

Chapter 2

Machine Learning and Its Applications

In this chapter, we provide a concise introduction to machine learning, laying the groundwork for subsequent chapters. Machine learning is a discipline in artificial intelligence (AI) that deals with data pattern extraction and recognition. It has applications in nearly all sectors, including healthcare, image processing, finance, and environmental sciences. As we outline the core terminology, we also illustrate the involved theory in two practical use cases: prostate cancer upgrading risk assessment and mosaic image segmentation. These use cases are based on our work previously published as [7] and [8], respectively. This chapter serves two main purposes: to provide the theory required to understand the methods for synthetic data generation we present in Chapter 4; and to illustrate what use cases of AI are common in practice, motivating why synthetic data (and privacy enhancing technologies in general) are important in the first place. Besides providing a concise introduction to AI, this investigation will therefore also answer the research question *What is machine learning and does the advent of its methods result in a need for reliable privacy enhancing technologies?*

2.1 Machine Learning Theory

2.1.1 Introduction

Machine learning (ML) is a subfield of artificial intelligence, often concerned with making inferences from large datasets [9]. While ML and data mining theory has a rich history dating back to the 1940s, ML algorithms have become an industry standard in recent decades [9]. Applications include recommender systems in retail, fraud detection in insurance, computer-aided diagnosis in healthcare, self-learning systems such as self-driving cars and more.

The field of ML can be subdivided into supervised and unsupervised learning, each associated with specific collections of problems, models, and algorithm classes. Intuitively put, supervised learning models have access to a dataset with dependent and independent variables, from which they infer patterns. When given independent variables from new data points, they can then apply these patterns to accurately predict the corresponding dependent variables. Supervised learning is further subdivided into classification, where the dependent variable is one of finitely many classes, and regression, where the dependent variable is real-valued. Various methods, including artificial neural networks, can be applied to both types of supervised learning problems.

In unsupervised learning, information is extracted from new datasets directly, without relying on patterns inferred from previously available data. The two most prominent examples of unsupervised learning are clustering and reinforcement learning. Clustering is the process of grouping together records in a database based on the degree to which they exhibit similar features. Reinforcement learning models how humans learn in an unsupervised manner, through interaction with their environments. An overview of the field of ML is provided in Figure 2.1.

In the remainder of this chapter, we introduce the core ML terminology applied throughout the thesis. This introduction is largely based on [9, 10, 11], to which the reader is directed for further reading. This terminology is also illustrated in a straight-forward example in Section 2.2 and in a more sophisticated use case in Section 2.4.

2.1.2 Basic Definitions

In this section, we consider tabular datasets. The definitions readily extend to other types of datasets. In particular, we will consider relational datasets in Chapter 4; terms are analogously redefined in said chapter.

A *dataset* D is a table, that is: a collection of rows $\{t_1, t_2, ..., t_n\} =: D$. A row t is a tuple defined over a set of attributes $A(D) = (a_1, ..., a_p)$. We let $V(D, a)$ denote the domain of attribute a in dataset D, with $v(t, a) \in V(D, a) \cup \{\varnothing\}$ the value of attribute a in row t and dataset D, for $\varnothing$ representing missing values (records for which the value of an attribute is unknown). The *type* of an attribute a determines the nature of its domain $V(D, a)$. We consider five cases:

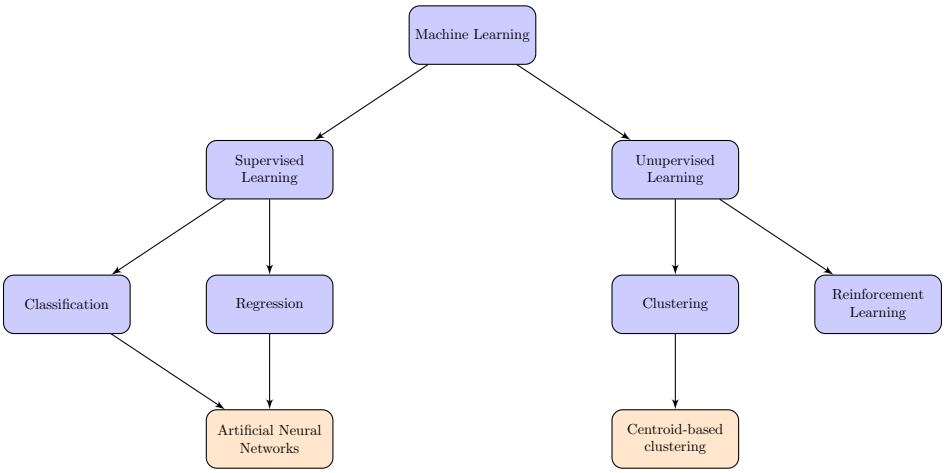


Figure 2.1: Basic ML taxonomy with some examples of problem classes and examples of solution algorithms (orange)

- a *real-valued* attribute has a domain $V(D, a) \subseteq \mathbb{R}$;
- a *discrete* attribute has a domain $V(D, a) \subseteq \mathbb{Z}$;
- a *time* attribute has a domain $V(D, a) \subseteq \mathbb{N}$ and its values represent time instants;
- a *categorical* attribute has a finite domain $V(D, a)$ consisting of non-numerical items and without a natural ordering;
- a *binary* attribute has a domain $V(D, a)$ with only two values.

We say that two datasets D, D' are *compatible* if $A(D) = A(D')$ and $V(D, a) = V(D', a)$ $\forall a \in A(D)$. We denote by $D \sim D'$ that D and D' are compatible.

2.1.3 Supervised Learning

A ML problem is a *supervised learning problem* if the following hold:

1. Given are two datasets D, D' such that $D \sim D'$;
2. The datasets D and D' have a *target attribute* $\hat{a}$;
3. The values $v(t, \hat{a})$ of $\hat{a}$ are known for all rows $t \in D$;
4. The objective is to predict the values $v(t, \hat{a})$ for rows $t \in D'$ by leveraging the information contained in D.

We will call a supervised learning problem a *regression problem* if the target attribute is real-valued, discrete or a time attribute. We call a supervised learning problem a *classification problem* otherwise.

A *model* is a mapping $m : \prod_{a \in A(D) \setminus \{\hat{a}\}} V(D, a) \rightarrow V(D, \hat{a})$. Given a supervised learning problem, a *solution* or *learning algorithm* is a procedure that, given dataset D and target column $\hat{a}$, produces a model m. Intuitively put, a model m is considered good if $m(t)$ closely approximates $v(t, \hat{a})$. For an overview of solutions for specific solutions for the most common ML problems, the reader can consult Table 2.1 and the references therein.

In practice, this is evaluated through *effectiveness metrics*. To do so, the dataset D is partitioned into a *training set* D_{train} and a *testing set* D_{test} (i.e. $D = D_{\text{train}} \cup D_{\text{test}}$ with $D_{\text{train}} \cap D_{\text{test}} = \varnothing$). The solution is applied to D_{train}, giving a model m. The model is then applied to attributes $A(D) \setminus \{\hat{a}\}$ of D_{test}. An *effectiveness metric* assigns a numerical value $e(m, D_{\text{test}})$ whose semantic is, without loss of generality, the greater, the better the model approximates the values $v(t, \hat{a})$ for $t \in D_{\text{test}}$.

A model can *overfit* to a particular dataset, that is: its solution may infer patterns that are overly specific to the training set, and do not generalize well to new data. To deal with this, models can be evaluated through cross-validation. This is a methodology in which the dataset D is partitioned into subsets of equal cardinality, i.e. into sets $D_1, D_2, ..., D_k$ with $D_i \cap D_j = \varnothing \ \forall i \neq j$; $\bigcup_{i=1}^{k} D_i = D$; $|D_i| = |D_j| \ \forall i, j$. Training is then reiterated k times, with each of the sets D_i for $i = 1, 2, ..., k$ the test set in exactly one iteration. By using a different test set in each iteration, and updating the model accordingly, the risk of overfitting is significantly reduced.

We denote by $m_{\boldsymbol{\theta}}$ a model dependent on *model parameters* $\boldsymbol{\theta} \in \boldsymbol{\theta}$, where $\boldsymbol{\theta}$ is the parameter domain. A procedure is then synonymous with *model training* (model fitting): identifying the parameters $\boldsymbol{\theta}^*$ optimizing a function of effectiveness metrics. Besides model parameters, a model may also depend on hyperparameters. These are parameters specific to classes of models that cannot be inferred from the data. Finding an optimal model therefore requires finding both the optimal model parameters and the optimal hyperparameters. For each combination of hyperparameters, the corresponding optimal model parameters have to be identified through a separate round of model training. *Hyperparameter optimization* or tuning is done through (meta)heuristics, exact methods or by informed guesses by the modeler and typically requires considerable computational resources.

Example 2.1.1 *An insurance company has extensive client records, including the age, income, occupation, gender, etc., of its clients. For a large collection of records, it has been verified whether or not the involved client exhibited fraudulent behavior. The insurer wants to use this information to predict whether new clients are likely to engage in fraudulent behavior.*

This is a classical example of a (supervised) classification problem: the insurer wants to classify clients as either fraudulent or not. To do so, they can subdivide the data for which the fraud-status is known into a training and a

testing set. Subsequently, they can train a model on the training set and use effectiveness metrics to evaluate it on the testing set. If the model showcases a strong degree of effectiveness, it can reliably be used to classify (potential) clients for which the fraud-status is unknown.

Example 2.1.2 *A hospital maintains a large database of patient records. Various diseases are difficult for physicians to diagnose, as their symptoms present similarly, are subject to many exceptions, and vary depending on underlying health conditions and prior medications. The hospital wants to leverage their full patient database to assist physicians in their diagnostic practices.*

This again is a classification problem. The hospital can leverage patient records with correct diagnoses (verified in retrospect) to train a model that assigns the correct diagnosis to new patients with a quantifiable high degree of accuracy. This model can infer all exceptions, including influences of underlying health conditions and prior medication.

2.1.4 Unsupervised Learning

Unsupervised learning is any type of ML problem or solution in which there is no training dataset available. As such, there is no dependent variable and methods are applied directly to the harvested data. In this thesis, we are mostly concerned with supervised learning problems. However, clustering plays a significant role in Chapters 4 and 5. The specific problem, solution and effectiveness metrics used in these chapters are therefore discussed here.

An ML problem is a *clustering problem* if the following hold:

1. Given is a dataset D with attributes $A(D)$;
2. The objective is to find a partition $\mathcal{D} = \{D_1, D_2, ..., D_k\}$ of D ($\bigcup \mathcal{D} = D$; $D_i \cap D_j = \varnothing \quad \forall i \neq j$) such that if row t is assigned to subset D_i, then it resembles some row $t' \in D_i$ more closely than it resembles $t'' \in D_j$ for any $j \neq i$

A *solution* to a clustering problem is a method that returns a partition $\mathcal{D}$ of dataset D. An *effectiveness metric* is a mapping that assigns a numerical value $e(\mathcal{D}) \in \mathbb{R}$ to a partition $\mathcal{D}$.

Both solutions and effectiveness metrics typically rely on distance metrics defined on the space $A(D)$. In Chapters 4 and 5, we will use a centroid-based clustering method. Such methods typically proceed along the following steps:

1. Select k rows $t_1, ..., t_k$ at random; put $\mu_i = t_i, \quad i = 1, 2, ..., k$
2. For each row $t \in D$, compute $l^* = \arg\min_{i \in \{1,2,...,k\}} d(t, \mu_i)$, for $d : A(D) \times A(D) \to \mathbb{R}$ a distance measure on $A(D)$;
3. Put $D_{l^*} \leftarrow D_{l^*} \cup \{t\}$;
4. Put $\mu_i \leftarrow M(D_i)$, for M some measure of centrality;

5. Repeat steps 2,3 and 4 until the values μ_i remain unaltered in a given iteration.

The number k of clusters is a hyperparameter of this method. If $V(D, a) \subseteq \mathbb{R} \quad \forall a$, we may use the geometric mean as a measure of centrality, that is: $M(D_i) = \frac{1}{|D_i|} \sum_{t \in D_i} t$. In this case, we refer to the method as k-means clustering.

An *effectiveness metric* assigns a numerical value $e(\mathcal{D})$ whose semantic is, without loss of generality, the greater, the more accurate the clustering policy. Intuitively put, a clustering policy is good if there is a high degree of similarity between records in a cluster, and a low degree between records in different clusters.

Example 2.1.3 *A social media platform has amassed a large amount of user data. A vendor of clothing that are tailored to a specific youth culture wants to run advertisements on this platform. An ad is considered effective when it is clicked, that is: when the platform's user goes directly from the social media platform to the vendor's website. An ad is ineffective if it is shown to a user who has no subsequent interaction with the vendor.*

While the vendor wishes to maximize the number of effective ads, the social media platform has incentives to minimize the number of ineffective ads. This is because ineffective ads take up airtime which could have been dedicated to more effective ads. Moreover, ineffective ads lead to a less immersive user-experience. Thus, the vendor is offered fixed unit of advertisement time and space, but wishes that these are used in the most effective way possible.

As youth cultures are defined by proximity in social factors, such as clothing, music, film and literature taste, age, gender, socio-economic status, etc., the platform can perform a clustering method based on such characteristics. At least one of the resulting clusters will then correspond most closely to the clothing vendor's target demographic. By showing the ads to this/these cluster(s), the platform can optimize the number of website visits, while simultaneously minimizing the number of ineffective ads.

Other unsupervised learning problems include reinforcement learning (RL) [12] and dimensionality reduction (DR) [13]. RL is a field of ML that models the manner in which individuals learn from experience, through penalties and rewards. In the resulting models, *agents* interact with a probabilistic *environment* or *state-space*. Their *actions* result in a direct *pay-off* (either positive or negative), and a transition to a new state. The agents aim to optimize a function of the obtained pay-offs (typically a discounted sum in which more recently obtained rewards are given exponentially more weight). By doing so, they obtain an optimal manner in which to interact with their environment. RL is closely related to the study of Markov Chains and Markov Decision Processes (see, e.g. [14]).

DR is a branch of ML that is typically tasked with data compression and noise reduction. DR has its roots in information theory, where the smallest number of bits needed to convey a piece of information was studied. Among the most

Problem	Supervised?	Applications	Algorithms/models	Reference(s)
Classification	Yes	Fraud detection; Client segmentation; Medical diagnosis; e-Mail spam filtering; Recommender systems.	Naive Bayes; Decision tree; Random forest; Logistic regression; Adaptive boosting; Support vector machine; Artificial Neural Network;	[9, 10, 15] [9, 10] [9, 10, 16] [9, 10] [9, 10, 17] [10, 15] [11, 10]
Regression	Yes	Financial forecasting; Pattern recognition; Process optimization.	Linear regression; Logistic regression; Ridge regression; Support vector machine; Artificial Neural Network;	[9, 10] [9, 10] [9, 10] [10, 15] [11, 10]
Clustering	No	Client segmentation; Recommender systems; Market basket analysis; Social network analysis.	Centroid-based clustering; Hierarchical clustering; Density-based clustering; Partition-based clustering;	[9, 10, 15] [9, 10, 15] [9, 10] [9, 10]
Reinforcement learning	No	Self-learning systems; Video game opponent modeling; Self-driving vehicles.	SARSA; Q-learning; Value/policy improvement algorithms.	[12] [12] [12] [12]
Dimensionality reduction	No	Data (including image) compression	Principal Component Anlysis (PCA) Autoencoders	[13]

Table 2.1: Concise overview of prominent ML problems, applications and solution methods

prominent DR methods at present is principal component analysis (PCA) [13]. This technique is has its roots in linear algebra and is applied for both dimensionality and noise reduction. In essence, PCA and other DR methods take a dataset as input. Subsequently, they learn to represent this set in a lower-dimensional space. This reduces the total amount of information, while preserving all the valuable structures. In noise removal applications, the methods also discern useful information from random noise. The lower-dimensional data representation is often referred to as a *code*, making the compression method an *autoencoder*.

2.1.5 Effectiveness Metrics

Each problem class has distinct effectiveness metrics for assessing model accuracy. For classification problems, efficiency metrics assign a numeric value $e(m, D_{\text{test}})$ to a model m applied to test set $D_{\text{test}} \subseteq D$. While the values of target attribute $\hat{a}$ are known for all rows $t \in D$, model m is applied to the remaining attributes $A(D) \setminus \{\hat{a}\}$. The predicted values $m(t)$ of target attribute $\hat{a}$ can then be compared to their real values $v(t, \hat{a})$ for all $t \in D$, providing in indication of how well the model predicts unforeseen test data. For clustering problems, metrics assign a real number $e(\mathcal{D})$ to the partition $\mathcal{D}$ of dataset D, quantifying the internal cohesion or external dissension of each cluster.

Classification Problems

Consider a classification problem with n distinct classes, a model m and a test set D_{test} and classes $V(D, \hat{a}) = \{c_1, c_2, ..., c_k\}$. The *confusion matrix* of m and D_{test} is an $n \times n$ matrix $M^{\text{conf}}(m, D_{\text{test}})$, such that

$$M^{\text{conf}}_{i,j}(m, D_{\text{test}}) = |\{t \in D_{\text{test}} : m(t) = c_i \wedge v(t, \hat{a}) = c_j\}|$$

The entry on the main diagonal of a confusion matrix indicate how many instances model m correctly classified. If $n = 2$, we refer to the problem as a *binary classification problem*. For such problems, there exists a bijection f from $V(D, \hat{a})$ to the set $\{0, 1\}$. We will wlog refer to $f^{-1}(0)$ as "positive" and to $f^{-1}(1)$ as "negative". The entries $M^{\text{conf}}_{1,1}(m, D_{\text{test}})$, $M^{\text{conf}}_{2,1}$, $M^{\text{conf}}_{1,2}$,$M^{\text{conf}}_{2,2}$ are respectively referred to as *true positive, false negative, false positive* and *true negative*. This is visualized in Table 2.2.

Often in binary classification problems, "positive" and "negative" have a semantics that favors methods that are overestimate positive cases over one that underestimates them. This is for instance the case in Example 2.1.2, where the hospital would rather have a system that incorrectly classifies healthy patients as unhealthy than misclassifying the other way around.

	$v(t, \hat{a}) = 0$	$v(t, \hat{a}) = 1$
$m(t) = 0$	True positive	False positive
$m(t) = 1$	False negative	True negative

Table 2.2: Confusion matrix for a binary classification problem

The most prominent metrics for classification problems are derived from confusion matrices. We will describe these wlog for binary classification problems.

Accuracy The accuracy, denoted by Acc, is simply the proportion of correct predictions, as formalized in equation (2.1).

$$\text{Acc}(m, D_{\text{test}}) = \frac{\text{Tr}(M^{\text{conf}}(m, D_{\text{test}}))}{\sum_i \sum_j M^{\text{conf}}_{i,j}(m, D_{\text{test}})} \tag{2.1}$$

In the binary case, equation (2.1) simplifies to equation (2.2).

$$\begin{aligned} \text{Acc}(m, D_{\text{test}}) &= \frac{M^{\text{conf}}_{1,1}(m, D_{\text{test}}) + M^{\text{conf}}_{2,2}(m, D_{\text{test}})}{M^{\text{conf}}_{1,1}(m, D_{\text{test}}) + M^{\text{conf}}_{1,2}(m, D_{\text{test}}) + M^{\text{conf}}_{2,1}(m, D_{\text{test}}) + M^{\text{conf}}_{2,2}(m, D_{\text{test}})} \\ &= \frac{\text{True positive + True negative}}{\text{True positive + True negative + False positive + False negative}} \end{aligned} \tag{2.2}$$

Precision Given a binary classification problem, a model m, test set D_{test} and the corresponding confusion matrix $M^{\text{conf}}(m, D_{\text{test}})$, we define the *precision*,

denoted by Prec according to equation (2.3).

$$\begin{aligned} \text{Prec}(m, D_{\text{test}}) &= \frac{M^{\text{conf}}_{1,1}(m,D_{\text{test}})}{M^{\text{conf}}_{1,1}(m,D_{\text{test}})+M^{\text{conf}}_{1,2}(m,D_{\text{test}})} \\ &= \frac{\text{True positive}}{\text{True positive + False positive}} \end{aligned} \tag{2.3}$$

In words, the precision is the proportion of instances classified as positive by the model, that are indeed positive in the real test data.

Recall Given a binary classification problem, a model m, test set D_{test} and the corresponding confusion matrix $M^{\text{conf}}(m, D_{\text{test}})$, we define the *recall* (also known as the true positive rate), denoted by Rec according to equation (2.4).

$$\begin{aligned} \text{Rec}(m, D_{\text{test}}) &= \frac{M^{\text{conf}}_{1,1}(m,D_{\text{test}})}{M^{\text{conf}}_{1,1}(m,D_{\text{test}})+M^{\text{conf}}_{2,1}(m,D_{\text{test}})} \\ &= \frac{\text{True positive}}{\text{True positive + False negative}} \end{aligned} \tag{2.4}$$

In words, the recall is the proportion of real positive test data that are classified as positive by the model.

F-Measure Given a binary classification problem, a model m, test set D_{test} and the corresponding confusion matrix $M^{\text{conf}}(m, D_{\text{test}})$, we define the *F-measure* (also known as the F-1 score), denoted by Fm, through (2.5).

$$\text{Fm}(m, D_{\text{test}}) = 2 \cdot \frac{\text{Rec}(m,D_{\text{test}})\cdot\text{Prec}(m,D_{\text{test}})}{\text{Rec}(m,D_{\text{test}})+\text{Prec}(m,D_{\text{test}})} \tag{2.5}$$

In words, the F-measure is the harmonic mean of the precision and recall.

Receiver Operating Characteristic (ROC) Consider a binary classification problem, a model m, test set D_{test} and the corresponding confusion matrix $M^{\text{conf}}(m, D_{\text{test}})$. Define the *false positive rate*, denoted by FPR according to equation (2.6).

$$\begin{aligned} \text{FPR}(m, D_{\text{test}}) &= \frac{M^{\text{conf}}_{1,2}(m,D_{\text{test}})}{M^{\text{conf}}_{1,2}(m,D_{\text{test}})+M^{\text{conf}}_{2,2}(m,D_{\text{test}})} \\ &= \frac{\text{False positive}}{\text{False positive + True negative}} \end{aligned} \tag{2.6}$$

The receiver operator characteristic (ROC) is an effectivieness metric that is particularly useful when the model m is a composite of two functions such that the first gives a real-valued output and the latter converts this into a discrete (e.g. binary) classification through hyperparameters, i.e.:

$$m(t) = m_2(m_1(t), \tau)$$

Where

$$m_1 : \Pi_{a \in A(D)\setminus\{\hat{a}\}} V(D, a) \to \mathbb{R}; \quad m_2 : \mathbb{R} \times X_\tau \to V(D, \hat{a})$$

for X_τ a domain of hyperparameters (note that these must also be inputs of the composite function m, its domain thereby becoming $\Pi_{a \in A(D) \setminus \{\hat{a}\}} V(D, a) \times X_\tau$). This is for instance the case if regression methods are applied to classification tasks: the regression model m_1 returns a number $m_1(t) \in \mathbb{R}$ for a given row t. The threshold function m_2 then converts m_1 to the most suitable class $m_2(m_1(t), \tau)$ through hyperparameters $\tau \in X_\tau$. In a binary classification problem, with a single real-valued threshold parameter, this latter function can for instance be of the form

$$m_2(x, \tau) = \begin{cases} 0, & \text{if } x \leq \tau \\ 1, & \text{otherwise} \end{cases} \tag{2.7}$$

for a given threshold $\tau \in \mathbb{R}$.

The dependency of the model on hyperparameters X_τ entails that the effectiveness is also affected by parameters $\tau \in X_\tau$. Assume that $X_\tau = \mathbb{R}$, and m_2 is of the form of equation 2.7. Then, as τ increases, the number of positive classifications (rows t with $m(t, \tau) = 0$) increases. Thus, all else being equal, the false positive rate is a monotone increasing function of τ. Likewise, the recall is a monotone increasing function of τ. The rate at which these metrics increase, however, may differ.

The *receiver operating characteristic (ROC)* curve is the curve obtained when plotting the increase in the recall as a function of the increase in the false positive rate. A large area under the ROC curve indicates that the recall increases significantly faster than the false positive rate. Thus, even for relatively small threshold values, we can identify true positives without incorrectly classifying a large number of negatives. This is why the area under the ROC curve, denoted by AUC, is used as an efficiency metric for classification problems.

Regression Problems

In regression problems, the objective is to find a models that capture the dependence of a real-valued dependent variable on the independent ones. The effectiveness of regression solutions is again quantified by comparing the output model's estimates $m(t)$ to real values $v(t, \hat{a})$ for rows $t \in D_{\text{test}} \subseteq D$.

Mean Squared Error Given a regression problem, a model m, test set D_{test}, the mean squared error, denoted by MSE, is computed according to equation (2.8).

$$\text{MSE}(m, D_{\text{test}}) = \frac{1}{|D_{\text{test}}|} \sum_{t \in D_{\text{test}}} \left(m(t) - v(t, \hat{a})\right)^2 \tag{2.8}$$

Coefficient of Determination Given a regression problem, a model m, test set D_{test}, the coefficient of determination, denoted by R^2, is computed according

to equation (2.9).

$$R^2(m, D_{\text{test}}) = 1 - \frac{\text{SS}_{\text{Res}}(m, D)}{\text{SS}_{\text{Tot}}(m, D)} \tag{2.9}$$

for

$$\text{SS}_{\text{Res}}(m, D) = \sum_{t \in D_{\text{test}}} \left(m(t) - v(t, \hat{a})\right)^2$$

and

$$\text{SS}_{\text{Tot}}(m, D) = \sum_{t \in D_{\text{test}}} \left(m(t) - \mu_{\hat{a}}\right)^2$$

with $\mu_{\hat{a}}$ the arithmetic mean of real values of attribute $\hat{a}$ in test set D_{test}. Note that SS_{Res} is simply the squared error, a measure indicating how much each model output deviates from its real value in the test set. The value SS_{Tot} measures how much each model output deviates from the true mean. The coefficient of determination, in combining these quantities provides an effectiveness metric that accounts for variance in the data.

That is: if there is considerable variance in the data, and this variance is accurately captured by the model, $\text{SS}_{\text{Tot}}(m, D)$ should be fairly large, while $\text{SS}_{\text{Res}}(m, D)$ should still be as small as possible, giving a large R^2 value. If there is little variance in the data, the degree to which individual, smaller fluctuations are accurately captured by the model is better measured due to the small value in the denominator in equation (2.9)

Clustering Problems

Recall that the output of a clustering method is a partition $\mathcal{D}$ of the dataset D. An effectiveness metric assigns a numerical value $e(\mathcal{D})$ to such partitions. Such a metric quantifies how well the algorithm groups together similar objects and the extent to which distinct groups are dissimilar enough to be considered as such.

Sum of Squared Errors Given is a partition $\mathcal{D}$ of a dataset D. The *sum of squared errors* of $\mathcal{D}$, denoted by $\text{SSE}(D)$, is defined through equation (2.10), where $\mu_{D'}$ is a centroid of D' and d is a distance metric on $A(D)$. For example, if $A(D) \subseteq \mathbb{R}^n$ for some $n \in \mathbb{N}$, then the standard arithmetic mean $\frac{1}{|D'|}\sum_{t \in D'} t$ can be chosen for computing $\mu_{D'}$ and the standard Pythagorean distance can be chosen for d.

$$\text{SSE}(\mathcal{D}) = \sum_{D' \in \mathcal{D}} \left[\sum_{t \in D'} d(t, \mu_{D'})^2\right] \tag{2.10}$$

Note that in equation (2.10), the inner sum is simply the variance within cluster D'. Thus, the sum of squared errors is the sum of the variances within the individual clusters.

2.2 Case Study: Prostate Cancer Upgrading Prediction

This section is based on [7] and provides a clear example of how the outlined theory is applied in practice. The importance of the results, directly affecting patients' life qualities, also exemplifies the strong need for AI methods in practice. We are given a dataset of 8357 patients who underwent robotic radical prostatectomy (RRP) and lymph node dissection at a single center between 2012 and 2017. The following attributes are available: age, body mass index, number of cores, number of positive cores, percentage of cancer, primary gleason, secondary gleason, total PSA, PSA density, ASA score, prostate volume, transitional zone volume and clinical stage.

Our objective is to build ML models to predict prostate cancer upgrading. We use three classification solutions to do this: a standard logistic regression model, a classification tree, and a random forest. We then evaluate the results of each using the AUC (area under the ROC curve). The results are also compared to standard monograms, as used by physicians in assessing prostate cancer upgrading.

The results show that all three methods outperform the standard monogram, with the random forest classifier showcasing the best results (AUC = 0.78), followed by the classification tree (AUC = 0.76) and the logistic regression model (AUC == 0.66). These results are visualized in Figure 2.2. Note that the curve of the random forest increases more rapidly then other curves, indicating that it can correctly clasify more positive instances with a lower number of false positives then the other models. This initial faster increase entails a larger AUC. The dotted line indicates the baseline, in which a fixed percentage of data points is assigned a positive classification at random.

We note that the results seem to improve with the complexity of the model, with the random forest classifier significantly outperforming the other classifiers. Furthermore, all three methods drastically outperformed the standard monograms used in practice by medical experts during diagnosis. This indicates that advanced AI can significantly outperform legacy healthcare methods. Unfortunately, such methods have to be implemented by experts and require access to personal data. Traditional anonymization methods take a long time to be applied, as do data privacy protocol clearances, resulting in long data lead times. Furthermore, these methods come with specific caveats and shortcomings. In consequence, synthetic data generation can potentially become a disruptive technology in the medical sector.

2.3 Artificial Neural Networks

2.3.1 Perceptrons

A *perceptron* is the basic unit of any artificial neural network (aNN). In this section, we describe how perceptrons can be applied to binary classification

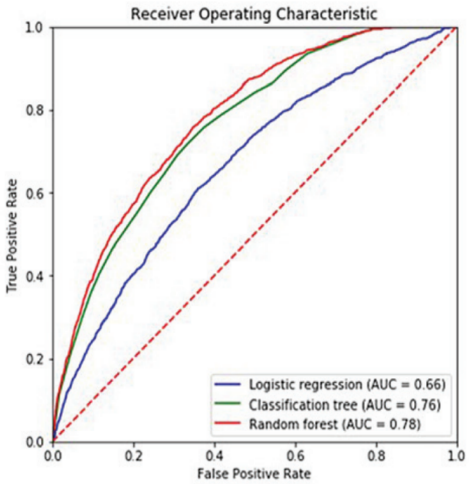


Figure 2.2: ROC curves of the learning methods

problems to exemplify the underlying theory. Perceptrons and aNNs can, however, also be applied to may other ML problems.

In this context, perceptrons are a class of models that return a binary classification when given a row of real-valued attributes. Let D be a dataset with $A(D)\setminus\{\hat{a}\} \subseteq \mathbb{R}^n$ for some $n \in \mathbb{N}$. A perceptron m is a model that returns a binary classification $m(t) \in \{0,1\}$ for a row $t \in A(D)\setminus\{\hat{a}\}$. The mapping m is a composition of two functions: a hyperplane function and an *activation function* ψ. More formally put:

$$m(t) = (\psi \circ m')(t)$$

were

$$m'(t) = \boldsymbol{\theta}^T t$$

for a vector $\boldsymbol{\theta} \in \mathbb{R}^n$ and ψ is a given and fixed activation function. The perceptron model m is completely determined by the parameters in vector $\boldsymbol{\theta}$, the form of the activation function being a hyperparameter. We denote by $m_{\boldsymbol{\theta}}$ the perceptron model with parameter vector $\boldsymbol{\theta}$. Training a perceptron corresponds to finding the vector $\boldsymbol{\theta}$ such that $m_{\boldsymbol{\theta}}$ best separates the positive and negative instances in the training set.

Perceptrons linearly subdivide the data space into two halfspaces of the respective forms

$$H_1 = \left\{t \in \mathbb{R}^n : \boldsymbol{\theta}^T t > b\right\}; \quad H_2 = \left\{t \in \mathbb{R}^n : \boldsymbol{\theta}^T t \leq b\right\}$$

for $b \in \mathbb{R}$ a hyperparameter. By taking $\boldsymbol{\theta}' = (-1|\boldsymbol{\theta})$ and $t' = (b|t)$, with "|" denoting concatenation of a scalar to a vector, the more common writing format

$$H_1 = \left\{t \in \mathbb{R}^n : \boldsymbol{\theta}'^T t' > 0\right\}; \quad H_2 = \left\{t \in \mathbb{R}^n : \boldsymbol{\theta}'^T t' \leq 0\right\}$$

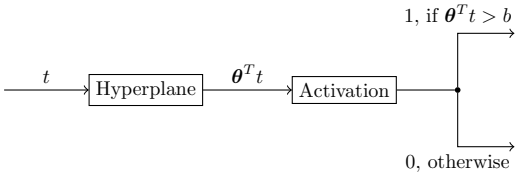


Figure 2.3: Structure of a perceptron with a discrete activation function

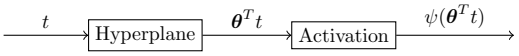


Figure 2.4: Generic structure of a single perceptron

is obtained. Note that the perceptron models become increasingly inaccurate the less evidently the data can be linearly separated by a single hyperplane.

The conceptually most straight-forward activation function is the to simply put

$$\psi(x) = \begin{cases} 1, & \text{if } x > 0 \\ 0, & \text{Otherwise} \end{cases}$$

This activation function divides the data space up into half spaces H_1, H_2 and classifies all data points in H_1 as positive and all points in H_2 as negative. A perceptron of this format is depicted in Figure 2.3. This choice of ψ is, however, discontinuous and hence not differentiable. This results in problems during model training. Continuous approximations are therefore often taken instead, most notably the *sigmoid function*

$$\psi(x) = \frac{1}{1 + e^{-x}}$$

Figure 2.4 shows a perceptron with generic activation function.

2.3.2 Artificial Neural Network models

Essentially, artificial neural networks (aNN) are networks in which nodes correspond to perceptrons or other data data transformation functions. A plethora of specific aNN model classes exist, with many developed with particular applications in mind. In this section, we introduce the most common types of aNN. In Section 2.4, a more specific class of aNN is described and applied in a practical case study. The study of aNN is a prominent subfield of *deep learning*.

Multi-layer Perceptrons *Multi-layer perceptrons* are layered networks in which each node is a perceptron. As a single perceptron classifies based on hyperplanes, a multi-layer perceptron network can classify based on convex polytopes (by definition intersections of halfspaces).

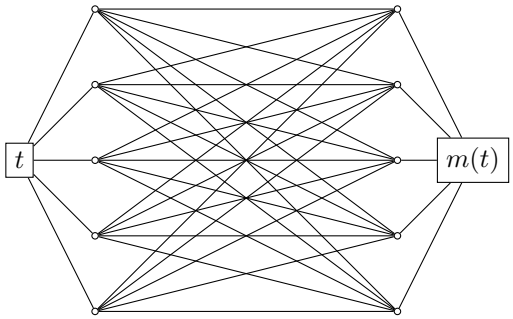


Figure 2.5: Multi-layer perceptron network (artificial neural network, aNN) with two layers plus the input and output layers.

A multi-layer perceptron can have any number of *layers*. Nodes in layer k have all the nodes in layer $k-1$ as their input. Hence a node i of layer k, denoted by $n_{k,i}$, is a perceptron that constructs a hyperplane based on the values of nodes in layer $k-1$, and applies its activation function to this hyperplane. The zeroth layer of the aNN is simply the input data, and the final layer are the output model values. This is illustrated in Figure 2.5.

Convolutional Neural Networks Convolutional neural networks (CNN), first introduced by LeCun [18], are layered networks of perceptrons in which at least one of the layers is a convolution operator. CNNs are particularly useful when the dataset D has a grid-like structure, for instance when the dataset is a set of images. As data goes through the convolutional layer, it is convolved with a filter, in place of the standard hyperplane computation. The output of such a convolutional layer is the activation function of each filter. In image processing, datasets are matrices (of color, saturation, hue, etc. values) rather than rows of attributes, making CNNs particularly applicable. Not all nodes in a given layer k necessarily take *all* outputs from preceding layer k as their input in CNN. In Section 2.4, a specific CNN framework is studied more closely, namely U-nets in the context of image segmentation.

Recurrent Neural Networks Recurrent neural networks (RNN), introduced by Rumelhart et al. [19], are layered networks of perceptrons in which some parameters are shared between multiple layers. RNNs are typically applied in the context of sequential data, for instance when the dataset D consists of time-series. Time series of the form $(x_1, x_2, ..., x_n)$ require that certain patterns are generalized over the entire sequence, such that the model can predict both individual sequence entries, but also properties shared between all entries in the sequence.

2.3.3 Training Artificial Neural Networks

Training a Single Perceptron

To train a single perceptron (that is: find the vector $\boldsymbol{\theta}$ such that the hyperplane $\boldsymbol{\theta}^T x$ most accurately divides the training data), we carry out the following steps.

Initialization The initial vector w must first be chosen. It is common to choose w randomly. This is for instance achieved by sampling from a uniform distribution in a range deemed suitable for the application at hand.

Learning The vector $\boldsymbol{\theta}$ is then incrementally updated to better represent the training data. To do so, a *loss function*, denoted by $L : \boldsymbol{\theta} \to \mathbb{R}$ is introduced. The loss function is a function that assigns a real-valued quantity $L(\boldsymbol{\theta})$ to measure how well the parameter vector $\boldsymbol{\theta}$ classifies training instances. The loss function compares model outputs $m_{\boldsymbol{\theta}}(t)$ to the corresponding actual values of attribute $\hat{a}$ of row t. In the context of classification, cross-entropy is the most commonly used loss function, defined through equation (2.11).

$$L(\boldsymbol{\theta}) = -\sum_{t \in D_{\text{train}}} (v(t, \hat{a}) \log(m_{\boldsymbol{\theta}}(t)) + (1 - v(t, \hat{a})) \log(1 - m_{\boldsymbol{\theta}}(t)) \tag{2.11}$$

Training data points $t \in D_{\text{test}}$ are provided as inputs to the perceptron on-by-one. In each step, weights $\boldsymbol{\theta}$ are then updated to minimize the loss function by a term $\Delta\boldsymbol{\theta}$, i.e.

$$\boldsymbol{\theta}_i \leftarrow \boldsymbol{\theta}_i + \Delta\boldsymbol{\theta}_i \quad \forall i$$

The value of $\Delta\boldsymbol{\theta}_i$ is obtained through gradient descent for the loss function. Note that we have to differentiate a composition of three functions to do so: the loss function, the activation function and the hyperplane computation. This is why the activation function is required to be differentiable.

Termination Training is complete when a set of stopping criteria is achieved. For example, training may terminate when each row of the training dataset has been used to improve the parameter vector.

Training artificial neural networks through backpropagation

Artificial neural networks are most commonly trained through (variations of) the *backpropagation algorithm*. The network is equipped with a loss function. Consider a network of k layers of l nodes each. Note that mathematically, the network is simply a composition of kl perceptrons (a hyperplane function and an activation function each, hence $2kl$ functions in total). All involved functions are differentiable, the gradient of the network can be computed simply through the chain rule for differentiation.

The backpropagation exploits this observation and is carried out as follows.

Forward pass Given a point $t \in D_{train}$, compute the output $m(t)$. To do so, simply apply all the functions of perceptrons in layer 1 of the network to the input, then use their outputs as the input for layer 2. repeat this until the output $m(t)$ of the final layer is obtained. For complex network topologies incorporating multiple computational steps, block diagrams (such as those in Figure 2.4 and 2.3, but for larger networks) can provide a visual aid in understanding how data flows through the network.

Backpropagation The output $m(t)$ and the true value of attribute $\hat{a}$ in row t are then available. To minimize the loss function, we can now use the chainrule, noting that each perceptron can be optimized individually. Thus, we can first use the output to train all perceptrons in final layer k. Once the involved parameters are optimized, we can proceed to layer $k-1$ and continue this process until we reach the first layer.

Termination Training is again complete if a set of stopping criteria is achieved. For example, training may terminate when all rows t of the test dataset D_{test} have been passed forward and subsequently backpropagated through the network. In practice, datasets can be too large to be entered into aNN implementations. The data is then subdivided into batches, and the training algorithm is applied to batches one by one. An *epoch* is then achieved once the network has trained using all batches once. Multiple epochs are often invoked during training, so that the model is informed by the full dataset multiple times.

2.4 Case Study: Mosaic image segmentation

This section is based on research previously published as [8].

2.4.1 Introduction

Cultural heritage is one of the most important assets of the society. Its preservation and restoration are time-consuming activities performed by experts and often consist in manual analysis of fine details of the works. It is hence natural that these tasks, as many others where human experts are involved in some form of data processing, are subjected to automation using machine learning techniques. Differently than other domains, however, tasks concerning cultural heritage may be harder because of the scarcity of labeled data and nature of the data itself. Despite these limitations, successful examples of applications exist, e.g., [20], and progresses in the techniques for different kinds of data pave the way for other successful applications.

We consider a particular kind of artistic works, namely mosaics. Mosaics are assemblies of small pieces of stone or similar materials, called *tiles* or tessellae, glued together with some binder or *filler*, such that the overall appearance of the assembly looks like a painting or some decorative pattern. Mosaics constitute an essential component of the cultural heritage for many (ancient) civilizations.

Preservation, and, to some degree, restoration of mosaics might be enhanced if digital versions of the works were available. Moreover, the access to the artistic works might be eased using digital means, possibly as part of a process in which hard copies are obtained starting from digital copies, hence enlarging the portion of population that can access mosaics, regardless of their physical location [21]. There have been a couple of approaches, namely [22, 23], that proposed automatic methods for obtaining a digital version of the mosaic. All of them take as input an image of the mosaic, that can be cheaply obtained also for non-relocatable mosaics, and output a *segmentation* of the image in which regions should correspond to tiles. Starting from the segmentation, a digital version of the mosaic may be obtained straightforwardly, hence easing the mosaic preservation and restoration and making it more accessible [24].

Here we propose a novel technique for mosaic image segmentation that is based on a recently proposed kind of convolutional neural networks (CNN), called U-net [25]. Our approach differs from the previous ones in the way the mosaic image is processed. The U-net processes the image at the pixel level, differently than the proposal by [23], but permits, by design, that some pixels are not associated with any region, differently than the approach of [22]: this means that using U-net for segmentation allows to model the presence of the filler. A key component of our approach is in the preprocessing phase that is part of the learning process: we propose a method for augmenting the dataset in such a way that the learning of U-net parameters is effective even when a small number of annotated examples are available. In facts, manual annotating mosaic images is a costly process [23].

We assess experimentally our approach applying it to 11 images of real mosaics, differing in style, age, and quality (both of the image and of the mosaic itself in terms of wear). We compare the segmentation based on U-net against previous methods using a set of established performance indexes suitable for the mosaic image segmentation task and we found that our method outperforms the other ones in the most relevant index. Moreover, we show that the way in which the three methods make errors in analyzing the image varies consistently with the fact that the methods are based on different underlying assumptions. This finding opens an opportunity for designing an even more effective method where U-net segmentation is a step of a more complex procedure which involves also other processing steps, eventually resulting in a better segmentation effectiveness.

2.4.2 Related works

In [22] the proposed approach aims to detect and to extract the tile from the filler, using the well-known watershed algorithm [26] and some mosaic-specific preprocessing. In [23] the authors proposal goal is the same, but they employed *deformable models* as flexible shapes to be superimposed on the mosaic picture and to be adapted to the effective shapes of the tiles. Other segmentation approaches include laser scanners and photogrammetry [27], segmentation based on already available mosaic cartoons [28]. We refer the reader to [29] for a

detailed review.

Regarding the U-nets, there are many applications in biomedical image segmentation, e.g., [30]. Variations of the U-nets have also been applied to volumetric segmentation from sparsely annotated volumetric images [31], road extraction from aerial images [32], and in case of ambiguous images, i.e., when many different annotations are available for every single image [33].

2.4.3 Problem statement

The goal of this work is to propose a method for segmenting an image of a mosaic in such a way that, for each tile of the mosaic in the image, all and only the corresponding pixels are assigned to the same region of the segmentation.

More formally, we call a *region* of the image I a subset of adjacent pixels of I. We call a *segmentation* of an image I a set $\mathcal{T} = \{T_1, \ldots, T_n\}$ of disjoint regions of I, i.e., $\forall i, j, T_i \cap T_j = \varnothing$.

Let $\mathcal{T}$ and $\mathcal{T}'$ be two segmentations of the same image I. Following [34], we adapt the previously outlined efficiency metrics for this particular case. We also add the so-called *count error*. This gives the following four indexes:

$$\mathrm{Prec}(\mathcal{T}, \mathcal{T}') = \frac{1}{|\mathcal{T}|} \sum_{T \in \mathcal{T}} \max_{T' \in \mathcal{T}'} \frac{|T \cap T'|}{|T|} \tag{2.12}$$

$$\mathrm{Rec}(\mathcal{T}, \mathcal{T}') = \frac{1}{|\mathcal{T}'|} \sum_{T' \in \mathcal{T}'} \max_{T \in \mathcal{T}} \frac{|T \cap T'|}{|T'|} \tag{2.13}$$

$$\mathrm{Fm}(\mathcal{T}, \mathcal{T}') = \frac{2|\mathcal{T} \cap \mathcal{T}'|}{|\mathcal{T}| + |\mathcal{T}'|} \tag{2.14}$$

$$\mathrm{Cnt}(\mathcal{T}, \mathcal{T}') = \frac{\mathrm{abs}(|\mathcal{T}'| - |\mathcal{T}|)}{|\mathcal{T}'|} \tag{2.15}$$

where $|T|$ is the number of pixels in the region T, $|\mathcal{T}|$ is the number of regions in the segmentation $\mathcal{T}$, and $T \cap T'$ is the set of pixels which belong to both T and T'.

The *precision* index $\mathrm{Prec}(\mathcal{T}, \mathcal{T}')$ is the average precision of regions in $\mathcal{T}$, where the precision of a region T is the largest ratio $\frac{|T \cap T'|}{|T|}$ among different $T' \in \mathcal{T}'$, i.e., the proportion of T pixels which belong to the region of $\mathcal{T}'$ with which T overlaps most. The *recall* index $\mathrm{Rec}(\mathcal{T}, \mathcal{T}')$ is the average recall of regions in $\mathcal{T}'$, where the recall of a region T' is the largest ratio $\frac{|T \cap T'|}{|T'|}$ among different $T \in \mathcal{T}$, i.e., the proportion of T' pixels which belong to the region of $\mathcal{T}$ with which T' overlaps most—it can be noted that $\mathrm{Prec}(\mathcal{T}, \mathcal{T}') = \mathrm{Rec}(\mathcal{T}', \mathcal{T})$. The *F-measure* (also known as F-1 score) is the harmonic mean of *precision* and *recall*. Finally, the *count error* index $\mathrm{Cnt}(\mathcal{T}, \mathcal{T}')$ is the normalized absolute difference between the number of regions in $\mathcal{T}'$ and the number of regions in $\mathcal{T}$.

It can be seen that, when the indexes are applied to the same segmentation, $\mathrm{Prec}(\mathcal{T}, \mathcal{T}) = 1$, $\mathrm{Rec}(\mathcal{T}, \mathcal{T}) = 1$, and $\mathrm{Cnt}(\mathcal{T}, \mathcal{T}) = 0$. Intuitively, the more similar the two segmentations $\mathcal{T}$ and $\mathcal{T}'$, the closer the precision and recall

indexes to 1 and the closer the count error index to 0. In the extreme case where $\mathcal{T} = \{I\}$, i.e., $\mathcal{T}$ consists of a single region covering the full image, recall is 1, whereas precision may be low and count error may be high; on the opposite case, if $\mathcal{T} = \{\{i\} : i \in I\}$, i.e., if regions of $\mathcal{T}$ correspond to single pixels of I, then precision is 1, recall may be low, and count error may be high.

Let $\mathcal{T}^\star$ be the unknown *desired segmentation* of a mosaic image I in which each region exactly corresponds to a tile in the image. The goal is to find a method that, for any image I of a mosaic, outputs a segmentation $\mathcal{T}$ which maximizes $\mathrm{Prec}(\mathcal{T}, \mathcal{T}^\star)$ and $\mathrm{Rec}(\mathcal{T}, \mathcal{T}^\star)$ and minimizes $\mathrm{Cnt}(\mathcal{T}, \mathcal{T}^\star)$.

2.4.4 U-net for mosaic segmentation

We propose a solution for the mosaic image segmentation problem described in the previous section which is based on a kind of Convolutional Neural Network (CNN). We assume that a *learning set* composed of images of mosaics and the corresponding desired segmentations are available. In a *learning phase*, to be performed just once, the learning set is used to learn the values of the parameters of the network. Then, once learned, the network is used in a procedure that can take any image I as input and outputs a segmentation $\mathcal{T}$.

The CNN used in this study is known as U-net, the name deriving from the shape of the ANN architecture. U-net was introduced by [25] who used it for the segmentation of neuronal structures in electron microscopic stacks: according to the cited study, U-net experimentally outperformed previous approaches.

When applied to an image, a U-net works as a binary classifier at the pixel level, i.e., it takes as input a 3-channels (RGB) image and returns as output a two-channels image. In the output image, the two channels correspond to the two classes and encodes, together, the fact that the pixel belongs or does not belong to the artifact of interest—in our case, a tile of the mosaic.

In order to obtain a segmentation from the output of the U-net, we (i) consider the single-channel image that is obtained by applying pixel-wise the softmax function to the two channels of the ANN output and considering just the first value, that we call the *pixel intensity* and denote by $p(i)$; (ii) compare each pixel intensity against a threshold τ; (iii) merge sets of adjacent pixels that exceed the threshold, hence obtaining connected regions. We discuss in detail this procedure in Section 2.4.4.

Internally, the U-net is organized as follows: a contracting path made of a series of 3×3 un-padded convolutions followed by max-pooling layers enables the context capturing while the expanding path consisting of transposed convolutions and cropping operations ensures precise features localization [25].

In our study we used an instance of the U-net tailored to input images of 400×400. In the contracting path, we used two 2-D un-padded convolutions steps of size 3, both made of 32 filters and followed by a rectified linear unit (reLU) precede a max-pooling layer with 2×2 pool-size. The same structure is repeated four times every time increasing the number of filters to 64, 128, 256, and 512. At the end of the contraction phase the 400×400 pixels input image in reshaped in a 17×512 tensor. In the expansion path, we started with an up-

sampling 2-D layer of 2×2 size of the features map followed by a concatenation with the correspondingly cropped feature map from the contracting phase and two 2-D un-padded convolutions steps of size 3×3 each with reLU activation function. The same procedure is repeated also four times every time reducing the number of convolutions filter by half leading to a tensor of shape 216×32. Furthermore a zero-padding 2-D layer reshapes the tensor in a $400 \times 400 \times 32$ shape prior to a 1-D convolution steps composed of two filters that gives in output a $400 \times 400 \times 2$ tensor that constitutes the output of the U-net. The output is then used to compute pixel intensities and hence the segmentation as briefly sketched above and detailed in Section 2.4.4.

Learning

Let $L = \{(I_1, \mathcal{T}_1^\star), \ldots, (I_m, \mathcal{T}_m^\star)\}$ be the learning set composed of m pairs, each consisting of a mosaic image I_i and the corresponding desired segmentation $\mathcal{T}_i^\star$, obtained by manual annotation. The outcome of the learning phase consists of the weights $\boldsymbol{\theta}$ of the U-net.

We first preprocess the pairs in the learning set L as follows, obtaining a different learning set L', for which $|L'| = |L|$ does not generally hold.

1. We rescale each pair $(I, \mathcal{T}^\star) \in L$ so as to obtain a given *tile density* $\rho_0 = \frac{|\mathcal{T}^\star|}{|I|}$, i.e., a given ratio between the number of tiles in the image and the image size; ρ_0 is a parameter of our method. We use a bicubic interpolation over 4×4 pixel neighborhood.

2. From each pair $(I, \mathcal{T}^\star) \in L$, we obtain a number of pairs by cropping square regions of I of size $l \times l$ (*crops*) that overlap for half of their size; l is a parameter of our method. Let $w \times h$ be the size of the image I of the pair, the number of pairs obtained by cropping is $\left(\left\lfloor \frac{w}{l} \right\rfloor + \left\lfloor \frac{w}{l} - \frac{1}{2} \right\rfloor\right)\left(\left\lfloor \frac{h}{l} \right\rfloor + \left\lfloor \frac{h}{l} - \frac{1}{2} \right\rfloor\right)$. We build a set L' including the resulting pairs, each one consisting of a square image of size $l \times l$ and a segmentation with, on average, approximately $\rho_0 l^2$ regions.

3. Finally, we augment L' by adding, for each of its pair, few pairs obtained by common image data augmentation techniques, i.e., rotation, horizontal and vertical flipping.

We remark that, when building L' from L, segmentations $\mathcal{T}^\star$ are processed accordingly to the processing of the corresponding images I.

In order to learn the weights $\boldsymbol{\theta}$ of the U-net, we consider a subset L'_{train} of L' that contains 90% of the pairs in L', chosen with uniform probability.

Then, we use the Adam optimizer [35] on image pairs in L'_{train} to learn the weights $\boldsymbol{\theta}$. We feed Adam with batches of 8 images and drive it by the following

weighted binary cross entropy loss function:

$$\text{Loss}(\boldsymbol{\theta}) = -\frac{1}{|L_\text{b}|}\sum_{I\in|L_\text{b}|}\sum_{i\in I}\Big(wq(i)\log p(i) + (1-q(i))\log(1-p(i))\Big) \tag{2.16}$$

where L_b is the batch of pairs $(I, \mathcal{T}^\star)$, w is a weighting factor, $p(i)$ is the pixel intensity of i obtained by applying the U-net with weights $\boldsymbol{\theta}$, and $q(i)$ is an indicator function that encodes in $\{1, 0\}$ the fact that the pixel i belongs or does not belong to a region of $\mathcal{T}^\star$:

$$q(i) = \begin{cases} 1 & \text{if } \exists T \in \mathcal{T}^\star, i \in T \\ 0 & \text{otherwise} \end{cases} \tag{2.17}$$

We use the weighting factor w in Equation (2.16) in order to weight differently classification errors for pixels of tiles and filler. The parameter w hence permits to cope with the fact that image of mosaics are in general highly unbalanced: much more pixel are associated to tiles than to the filler. We experimented with three different values of w: 0.5, 0.1, and 0.01, the former corresponding to weighting the two classes equally.

We set Adam to run the optimization for n_epoch epochs, with a learning rate that varies at every epoch using an exponential decay function. During the optimization, we use the remaining 10% of the set L' for monitoring the progress, by computing the loss of Equation (2.16) on $L' \setminus L'_\text{train}$.

Segmentation

In the segmentation phase, we use a learned U-net to obtain a segmentation $\mathcal{T}$ out of an image I, as follows.

1. We rescale the input image such that its estimated tile density ρ is approximately equal to the ρ_0 value used during the learning. For computing ρ, and hence for performing the scaling, we assume that a raw estimate of the number of tiles in the image I is available: in practice, this estimate might be obtained by visual inspection of a small portion of the image.

2. We apply the U-net to I obtaining a single-channel image of pixel intensities that we threshold at 0.5, hence obtaining a binary image of the same size of I. We call this image the *output mask*.

3. We consider the subset $I' = \{i \in I, p(i) \geq 0.5\}$ of pixels of I that are classified by the U-net as a belonging to tiles.

4. Finally, we partition I' in subsets composed of adjacent pixels, hence obtaining the segmentation $\mathcal{T} = \{T_1, T_2, \dots\}$.

2.4.5 Experimental evaluation

We performed an extensive experimental evaluation aimed at assessing our method effectiveness in terms of precision, recall, and count error on images of mosaics not used for learning. To this end, we considered a set of images of real mosaics, that we manually annotated to obtain the corresponding desired segmentations (i.e., the *ground-truth* segmentations), and applied our method.

We collected a dataset of 12 images of real mosaics including both images that we acquired with a consumer camera and images that we obtained online. Part of this dataset has already been used by Bartoli et al. [23].

The mosaics depicted in the images of our dataset belong to different ages in time and also differ in tile density and color. The annotation required for the training was performed manually. Despite the extensive effort and attention devoted to the process, some dissimilarities between a mosaic image and its corresponding ground-truth segmentation may still exist. Nonetheless, the manually annotated mask looks visually correct.

Figure 2.6 shows the images of our dataset. It can be seen that the images greatly vary in the density and size of the tiles as well as in the visually perceived sharpness of tile edges.

Procedure

We evaluated our method using a *leave-one-out* procedure on the images (and corresponding desired segmentations) of our dataset, as follows. For each pair $(I, \mathcal{T}^\star)$ in the dataset D, we (i) performed the learning on $L = D \setminus (I, \mathcal{T}^\star)$, hence obtaining a learned U-net, (ii) used the learned U-net for obtaining the segmentation $\mathcal{T}$ of I (i.e., the image of the left-out pair), and (iii) computed the precision $\mathrm{Prec}(\mathcal{T}, \mathcal{T}^\star)$, recall $\mathrm{Rec}(\mathcal{T}, \mathcal{T}^\star)$, and count error $\mathrm{Cnt}(\mathcal{T}, \mathcal{T}^\star)$.

Concerning the method parameters, we set $\rho_0 = 15 \times 10^{-5}$, $l = 400$, and $n_{\mathrm{epoch}} = 10$. We chose the values for ρ_0 and l based on the minimum dimension and tile density of the images in our dataset. In this way we obtained crops of 400×400 pixels with approximately $l^2 \rho_0 = 24$ tiles in each crop. In the segmentation phase, we set $\rho = \rho_0$ and computed ρ using the actual number $\mathcal{T}^\star$ of tiles.

We run the experiments using an implementation of the method based on Python 3.6 with Keras and Tensorflow; we executed it on some p3.8xlarge AWS EC2 instances, each equipped with 64 vCPU based on 2.3 GHz Intel Xeon E5-2686 v4 with 244 GB RAM and with 4 GPUs based on NVIDIA Tesla V100 with 32 GB RAM. In these settings, the learning time for one repetition of a leave-one-out procedure is 30 min and the segmentation time is in the order of few seconds.

Results and discussion

We show in Table 2.3 the results in terms of the salient segmentation effectiveness indexes presented in Section 4.3.2 (precision, recall, F-measure, and count error) for each mosaic image, i.e., each repetition of the leave-one-out procedure.

	U-net with $w = 0.5$				U-net with $w = 0.1$				U-net with $w = 0.01$			
Im. #	Cnt	Prec	Rec	Fm	Cnt	Prec	Rec	Fm	Cnt	Prec	Rec	Fm
0	0.49	0.56	0.71	0.62	0.18	0.62	0.54	0.57	0.19	0.74	0.40	0.51
1	0.52	0.57	0.69	0.62	0.20	0.70	0.56	0.61	0.19	0.81	0.30	0.43
2	0.75	0.41	0.76	0.51	0.40	0.51	0.52	0.51	0.40	0.51	0.40	0.43
3	0.18	0.64	0.73	0.68	0.07	0.79	0.67	0.72	0.17	0.96	0.35	0.51
4	0.15	0.66	0.61	0.64	0.10	0.68	0.57	0.62	0.08	0.79	0.38	0.51
5	0.31	0.66	0.62	0.64	0.26	0.68	0.57	0.62	0.21	0.86	0.23	0.36
6	0.32	0.58	0.61	0.59	0.17	0.63	0.60	0.61	0.11	0.81	0.32	0.46
7	0.30	0.67	0.80	0.73	0.26	0.73	0.76	0.75	0.41	0.73	0.70	0.71
8	0.28	0.59	0.71	0.64	0.29	0.62	0.67	0.64	0.27	0.68	0.53	0.59
9	0.21	0.62	0.73	0.67	0.23	0.67	0.60	0.62	0.13	0.73	0.52	0.60
10	0.52	0.65	0.70	0.67	0.41	0.70	0.62	0.65	0.31	0.88	0.31	0.45
11	0.29	0.65	0.72	0.69	0.25	0.68	0.68	0.68	0.24	0.83	0.23	0.35
Avg.	0.36	0.60	0.70	0.64	0.23	0.67	0.61	0.63	0.22	0.78	0.39	0.49

Table 2.3: Results obtained with the three U-nets with different w values.

Table 2.3 shows that average precision, recall, and F-measure at $w = 0.5$ are 0.60, 0.70, and 0.64 respectively, whereas the average count error is 0.36. By looking at the figures of single images, it can be seen that the effectiveness of segmentation varies among mosaic images, with the F-measure ranging from 0.51 of image 2 to 0.73 of image 7 and the count error ranging from 0.75 of image 2 to 0.15 of image 4. We carefully compared the numerical features of Table 2.3 with the corresponding mosaic images (see Figure 2.6) and found that the segmentation effectiveness is consistent with the subjectively perceived quality of the images. Good numerical results are obtained by our method for images 7 and 11, while the worst result is obtained for image 2 which exhibits poor sharpness.

Concerning the impact of the weighting parameter w, it can be seen from the three sections of Table 2.3 that it act consistently with its semantic. As w decreases, the balancing between precision and recall varies, namely precision increases and recall decreases: in facts, a lower value for w corresponds to a lower contribution, in the loss used during the learning (see Equation (2.16)), of the errors in classifying pixel belonging to the actual tiles. As a result, the learned U-net tends to outputs smaller regions that have a lower recall and a greater precision. Another effect is that the count error is lower with lower values of w, because there are fewer regions of the output segmentation in which tiles are "glued" together (see also later discussion). These finding on the impact of w on the output segmentations suggests that it can be used as a parameter to tailor the output to the specific usage intended by the user. However, since in our experiments $w = 0.5$ delivers the best F-measure, we report in the following only the results obtained in this settings.

In order to put our results in perspective, we compared them with those obtained by the two other existing methods for mosaic image segmentation, i.e.,

	U-net with $w = 0.5$				GA				TOS			
Im. #	Cnt	Prec	Rec	Fm	Cnt	Prec	Rec	Fm	Cnt	Prec	Rec	Fm
7	0.30	0.67	0.80	0.73	**0.03**	0.50	0.76	0.60	0.14	0.64	0.87	**0.74**
8	0.28	0.59	0.71	**0.64**	**0.03**	0.42	0.63	0.50	0.54	0.56	0.72	0.63
9	0.21	0.62	0.73	**0.67**	**0.01**	0.41	0.66	0.51	0.03	0.53	0.82	0.64
10	0.52	0.65	0.70	**0.67**	0.07	0.50	0.63	0.56	**0.06**	0.49	0.68	0.57
11	0.29	0.65	0.72	0.69	**0.03**	0.46	0.67	0.55	0.90	0.63	0.78	**0.70**
Avg.	0.32	0.64	0.73	0.68	0.03	0.46	0.67	0.54	0.33	0.57	0.77	0.66

Table 2.4: Results obtained with our method and with GA and TOS for a subset of the dataset.

[23] and [22], that we here denote by GA and TOS, respectively. Table 2.4 shows the values of the four indexes for the mosaic images of our dataset which were also processed with GA and TOS (for these methods, the figures are taken from [23]). For each image, we highlight in Table 2.4 the best Fm and Cnt figure among the three methods.

The foremost finding is that our method outperforms both GA and TOS in terms of average F-measure, with 0.67 vs. 0.54 and 0.66, respectively: considering Fm on the single images, our method obtains the best result in 3 on 5 images. Concerning the count error, U-net scores better than TOS (0.30 vs. 0.33) and worse than GA (0.30 vs. 0.03): we note, however, that the latter method is designed to output a number of tiles corresponding to the user-provided estimate.

Another finding concerns how the errors in segmentation are distributed between precision and recall. For all the three methods, recall is in general larger than precision, meaning that tiles in the computed segmentation are in general "larger" than the corresponding tiles in the desired segmentation. The unbalancing is, however, much greater in TOS and GA than in U-net, the difference between recall and precision being 0.09, 0.21, and 0.20 respectively for our method, GA and TOS. We think that this difference can be explained by the way the three methods work. In TOS, the segmentation does not allow to obtain regions which are not tiles: this means that the filler is always included in a tile, resulting in a low precision and good recall. In GA, the overlapping of tiles is not explicitly forbidden or discouraged, thus the precision is very low, on average, because the output segmentation often contains tiles which span across many desired tiles. In our method, instead, the network is trained to discriminate between pixels belonging or not belonging to a tile in the desired segmentation: the way the loss is computed during the training of the U-net (see Equation (2.16)) favors a good balancing between false positive and false negative classification at the level of pixels and, hence, between precision and recall.

In Figure 2.7 we compare the visual results of the segmentation of image 11 using the three methods. Due to the aforementioned differences between the algorithms, the number of tiles in the TOS segmentation is higher while the

size of the tiles tend to be smaller when compared to the other methods. In GA, since the algorithm allows for tiles overlapping, many of the predicted tiles share the same area. In the U-net segmentation some tiles are not properly separated, however position, size, and count are visually closer to those in the original image.

2.5 Conclusion

In this chapter, we introduced the main notions of machine learning. This includes the definitions of datasets and various types of data types; the generic problem structures of supervised learning methods (more specifically: classification and regression problems); a basic description of unsupervised learning with an emphasis on clustering; training, hyperparameter optimization, overfitting, cross-validation, and effectiveness metrics.

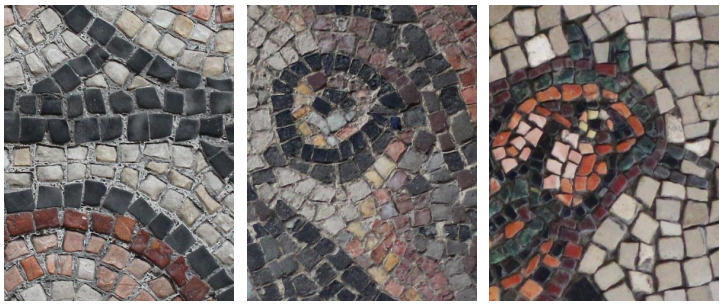
These concepts were illustrated in a case study, in which ML classification solutions were applied to an oncological real-world case. In this case study, we trained three classifiers to predict risk of prostate cancer upgrading. The results provide anecdotal evidence that the effectiveness of AI tasks increases with the sophistication of the involved models. As patient data is highly confidential, obtaining the data for the analysis was cumbersome. Combined with the direct impact of the results, this shows the need for privacy enhancement. Because direct data access was required to train the ML models, differential privacy (with queries returning aggregate results) was not an option. The fallibility of other traditional anonymization methods suggests that synthetic data would be the best option for secure and accurate analysis.

The core topics in deep-learning were also introduced. We covered the basic perceptron model, various kinds of neural networks composed of perceptrons and their training algorithms. We applied these notions to the problem of the segmentation of mosaic images, for which we proposed a method based on deep learning, namely U-net. We experimentally evaluated our proposal on a set of 11 images of real mosaics acquired in different conditions, with different image quality, and with different building properties. The results suggest that our method is effective, scoring the better value for the most relevant index on the majority of images used in the comparison.

We think that our results constitute a further evidence that modern deep learning systems can help solving tasks in a variety of fields, here in digital humanities. This further emphasizes the need for secure access to deep learning and artificial intelligence methods across industries and disciplines.

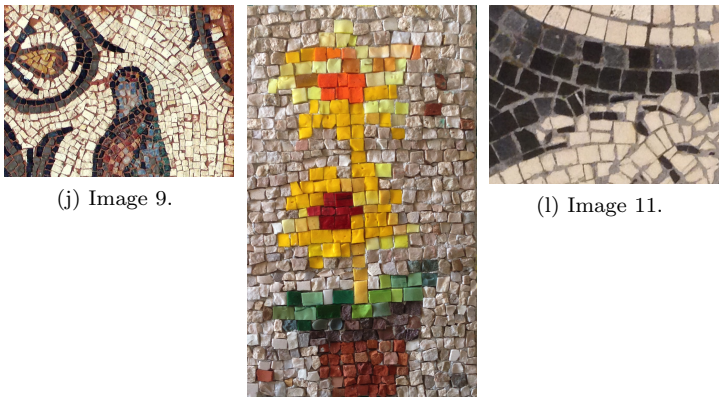
We believe that further improvements in mosaic image segmentation might be obtained. The most promising way to achieve them might be merging together two radically different techniques: the one presented in the present section, based on deep learning, and the one designed in [23], based on a different form of optimization which includes, in the solution presentation, some domain knowledge concerning the shape of the tiles.

(a) Image 0. (b) Image 1. (c) Image 2.



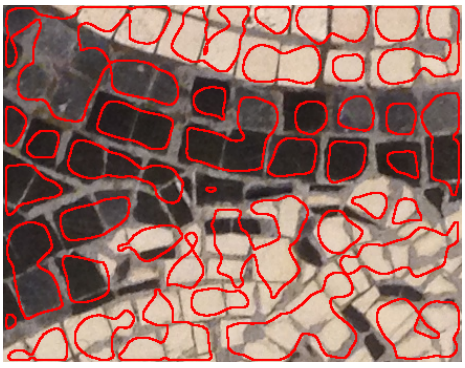
(d) Image 3. (e) Image 4. (f) Image 5.

(g) Image 6. (h) Image 7. (i) Image 8.

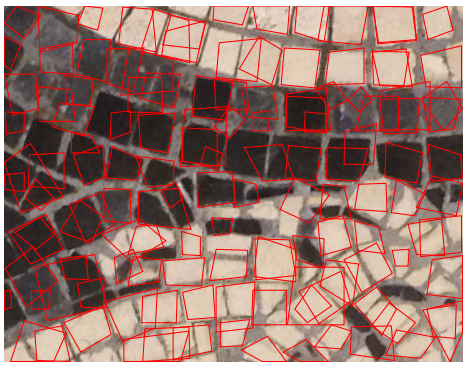


(j) Image 9. (k) Image 10. (l) Image 11.

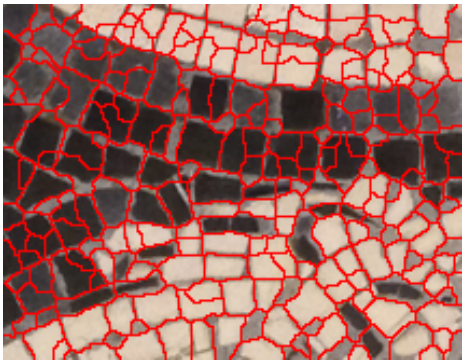
Figure 2.6: The images of the dataset. Images from 7 to 11 has been used also in [23, 34].



(a) U-net



(b) GA



(c) TOS

Figure 2.7: Example of segmentation of image 11 overlapped on the original image with the three methods.

Chapter 3

Data Privacy Law

In this chapter, we answer the research question: *under what conditions can datasets legally be considered void of sensitive information; which approaches exist to achieve these conditions; and how effective are these approaches at meeting the conditions?* To answer this question, we first outline the data privacy legislation effective in the Europe Union. We then provide an overview of privacy enhancing technologies, including a legal evaluation of their effectiveness. The discussion is motivated by practical examples of data breaches and legal cases. We find that of the existing approaches to data anonymity, synthetic data generation is the only one that can guarantee both data privacy and data utility. Due to the novelty of this approach, jurisprudence is still lacking. The topic is, however, gaining traction among legal scholars and policy-makers. In this chapeter, we use the terms *record*, *data point* and *datum* interchangeably to refer to the collection of attributes of a single unit in a dataset (a row in Chapter 2). A *data subject*, *individual* or *natural person* is the real-world individual to which a record refers. We will refer to the person or institution using data as the *data manager*. Section 3.2 is based on [36].

3.1 Data Privacy and Governance Legislation in the European Union

3.1.1 Legal Theory of Data Privacy

The General Data Protection Regulation (GDPR [37, 38]) came into effect in 2018. The core principle of this legislation is to protect the fundamental right of EU citizens over the processing of their personal data. This is further echoed by the European Commission (EC) strategy "Shaping Europe's Digital Future" [1]. At the same time, the "Towards A Common European Data Spaces" initiative will facilitate enhanced data exchange [2]. An important part of the initiative is the Data Governance Act, adopted in 2021 [39]. This act expressly states the need for data sharing.

The conflict between data protection on the one hand and the need for increased shared data spaces on the other is difficult to manage. Currently, Recital 26 of the GDPR states that it does not apply to anonymous information. Thus, the use of anonymous data could be a potential option in achieving both objectives. However, the definition of the term "anonymous" is subject to interpretation. The same GDPR article clarifies that anonymous information is deemed as such if it does not relate to "an identified or identifiable natural person". To be rendered anonymous, personal data shall be processed and used in such a manner that "the data subject is not or no longer identifiable" ([38]).

Neither the GDPR nor its predecessor, Directive 95/46/EC [40], clarify when data can be considered sufficiently anonymous. A working definition is, however, provided in Article 4 of Regulation EU 2016/679 [41]. This Article defines "personal data" in terms of twelve attribute categories that are lead to re-identification susceptibility (name, identification number, location data, online identifier, specific physical, physiological, genetic, biometric, mental, economic, cultural, and social identity data). Natural persons should be protected against both direct and indirect identification through these attribute classes. Globally, the most stringent data privacy legislation is the *Health Insurance Portability and Accountability Act* (HIPAA, [42]), applying to health insurance data in the United States solely. Like regulation EU 2016/679, the HIPAA defines "personal data" through a set of attributes that should be protected from identification. Where regulation EU 2016/679 identifies twelve such attributes, the HIPAA identifies seventeen, roughly covering the same categories as EU 2016/679, Article 4, but also including health plan membership, genetic test results of family members, manifestations of diseases or disorders in family members, requests for, or receipt of genetic services on the individual or family member. It also covers genetic test results of fetuses or embryos of pregnant individuals [43]. Neither regulation EU 2016/679 nor the HIPAA further specify how an indirect identification process may operate.

A more concrete conceptualization of (indirect) identification is provided by the "Article 29 Working Party on the Protection of Individuals with regard to the Processing of Personal Data" (WP29 [44]). In Opinion 03/2016 of the

WP29, three data attacks associated with data exchange are introduced, against which protection is required:

Re-identification attack A re-identification attack is an attack in which the attacker identifies an individual in the (anonymized) dataset. A re-identification attack is successful whenever an attacker can deduce the identity of at least one data subject. Re-identification attacks often identify records with exceptional combinations of attribute values, as these are easiest to single out. This is illustrated in Example 3.1.1. They may also rely on prior knowledge that the attacker, using known parameter values to infer unknown ones, as in Example 3.1.2.

Example 3.1.1 *Given is a dataset of patients of a medical condition that mainly affects the elderly. The attacker wants to infer the identity of at least one of the data subjects. However, the data managers use identification numbers in place of patients' names, hindering the attack.*

Because the condition is geriatric in nature, the younger the data subjects are, the more scarce their records become. The attacker can exploit this by targeting the youngest cohort in the dataset. This reduces the size of the initial target set of the attack considerably.

Next, the attacker can focus on another attribute for which some values are highly exceptional, for instance location. While most patients may live in large urban centers, only a select number may live in a particular rural area. Combined with the age requirement, this may single out particular individuals, culminating in a succesful re-identification attack.

Example 3.1.2 *Given is a dataset of clients of an insurance company. The dataset contains personal information such as age, gender, location, but also financial information such as income and premium. An attacker has an acquaintance who is a client of the insurance company. (S)He knows most of the acquaintance's personal information, but wants to know what hisher income is. If the data known to the attacker is sufficiently specific, (s)he can simply find the record matching the personal data and inspect the income attribute.*

Linkage attack A linkage attack is an attack in which the attacker links an individual's record in the (anonymized) dataset to another record belonging to the same individual. This other record may be in the same dataset, or a different available dataset. Linkage attacks are problematic, as they help may provide the attacker with sensitive information that can be directly linked to an individual data subject through an intermediary dataset. Example 3.1.3 illustrates how a linkage attack can take place.

Example 3.1.3 *Given is a publically available dataset D of a small family retailer containing customer preferences for particular ice cream flavors. This dataset is not considered confidential, as the contained data is innocuous. An*

insurance agency maintains a second database D' which contains detailed financial information of the clients, including their income levels. The datasets have multiple attributes in common: age, gender, etc. Contray to dataset D, dataset D' is considered confidential, hence numeric pseudonyms are used in place of names.

An attacker gains access to dataset D' and wants to identify the individual with the largest income. Fortunately, (s)he is unable to perform a successful re-identification attack. However, (s)he discovers the publically available dataset D. Through the overlap in attributes, the attacker can link the records in D to those in D', thereby identifying the targeted individual.

Parameter inference attack A parameter inference attack is an attack in which the attacker aims to infer the value of a parameter from the data. This type of attack provides the attacker with more specific information than generally available regarding a particular data subject, or a specific subset of subjects. Note that it is difficult to quantify the effectiveness of this type of attack, as it merely provides stronger inductive-statistical information (cf. an analogous theory of explanation by Hempel [45, 46]), rather than concrete and deterministic information. because of this, parameter inference attacks are rarely identified in practice. Hempel's [45, 46, 47] criteria for inductive-statistical inference may provide a formal framework to better identify when parameter inference attacks are conducted and the degree to which they are successful. Example 3.1.4 illustrates how a parameter inference attack can be conducted.

Example 3.1.4 *Consider again the premise of Example 3.1.2. This attack was successful if the combination of available personal information narrowed the dataset down to a single record. In practice, this may be rare. Suppose instead that the attacker's available information narrows the dataset down from* $20,000$ *records to* $2,000$ *(hence,* 10% *of the data). Suppose that in the overall dataset of* $20,000$ *data subjects, the median income is* $€25,319$ *(the Italian median income in 2019). However, in the subset of* $2,000$ *people meeting the attacker's conditions, the median income is* $€75,000$*. The attacker now knows that his acquaintance is highly likely to be of substantially above average wealth.*

Thus, GDPR compliance requires that data is sufficiently anonymous, so that none of these attacks are effective, regardless of their implementation. To render the attacks inapplicable, data needs to be sufficiently anonymized through some de-identification process prior to its transfer. The European Comission [44] expressly state that specific methods used to carry out attacks are constantly evolving and are of increasing technological sophistication. It is therefore impossible to give a concrete, exhaustive list of attack methodologies. In [48], specific anonymization techniques are discussed by the same Article 29 Working Party. These, among other techniques, are detailed in Section 3.2.

As a consequence, an anonymized dataset's ability to withstand attacks cannot be assessed using external means. For instance, when trying to deliberately conduct an attack in a controlled environment to measure data's level of security, experiments are limited to a given set of attack methodologies. Even if this

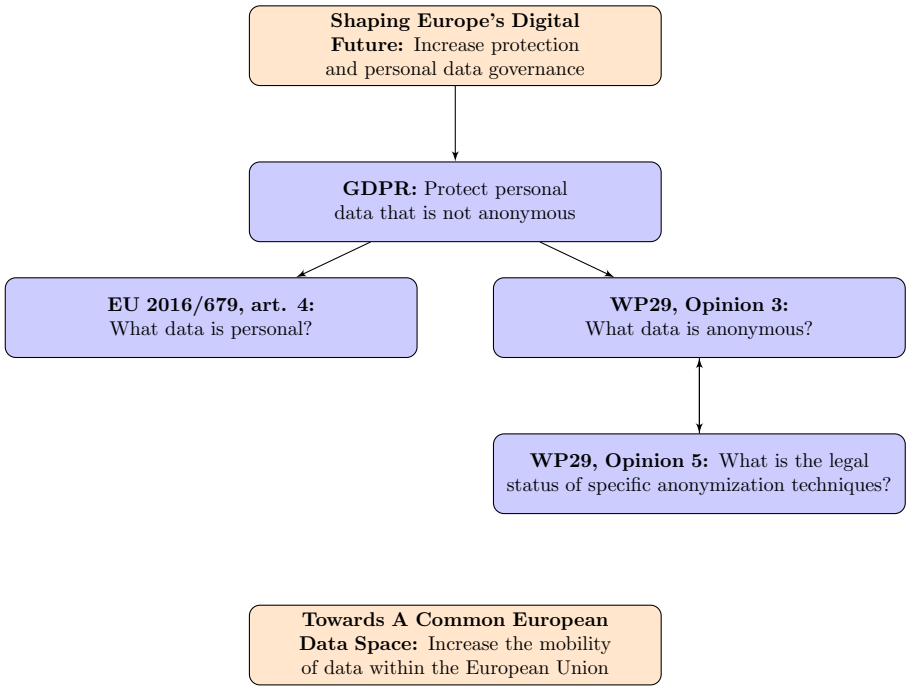


Figure 3.1: European Union legislation (purple) and European Commission objectives (orange) affecting data mobility.

set is representative of all classes of attack methods currently known, they may lack specific details and they may not be representative of all available methods after a period of time. Thus, evaluation of the effectiveness of anonymization techniques should include intrinsic metrics that infer security directly from data properties.

3.1.2 Data Breaches in Practice

As industry is rapidly incorporating big data and AI into their best practices, the number of annual data breaches is rapidly increasing. Since its implementation in May 2018, the number of issued GDPR fines has grown steadily to over 1100 fines, accounting for over $1.6 billion collectively in June 2022 [49]. Since its coming into force, GDPR fines have become more stringent as of January 2021, as is evident from Figure 3.2. In a 2021 survey, 94% of organizations reported to have had insider data breaches that year [50], with human errors being a common cause.

Re-identification attacks are among the most common attacks in practice. De Montjoye et al. [51] showed that four data points (location and time of the purchase using a credit card) are enough to uniquely re-identify 90 % of in-

dividuals. Indeed, many anecdotal evidences appear to confirm the issues of anonymization with respect to re-identification. In 2014, the New York City Taxi and Limousine Commission published an anonymised dataset with data of 173 million individual taxi trips. Hackers soon discovered that they could use the dataset to work out the incomes and addresses of the drivers [52].

Linkage attacks are also common in practice. In 1997, the Massachusetts Governor's medical records were identified by matching anonymous data from medical meetings with publicly available voter registration data [53]. In 2006, Netflix published part of its subscribers' viewing histories: it then turned out that users could be re-identified through Internet Movie Database (IMDb) data [54], forcing Netflix to remove the data mere days after the publication. A customer sued for breach of privacy; Netflix settled.

As mentioned, parameter inference attacks are harder to identify in practice. While information may be leaked, it is difficult to apply the moniker. In the 2014 NYC New York City Taxi and Limousine Commission case, some concerns over parameter inferences were raised alongside the re-identification concerns. Hern [52] points out that the dataset not only allows for the re-identification of drivers and their sensitive information. It can also provide information on the likely whereabouts of drivers and passengers. The degree to which this is a substantial problem are, however, not addressed. This underscores the lack of a framework for qualifying parameter inference attacks.

3.2 Data Anonymization Technology

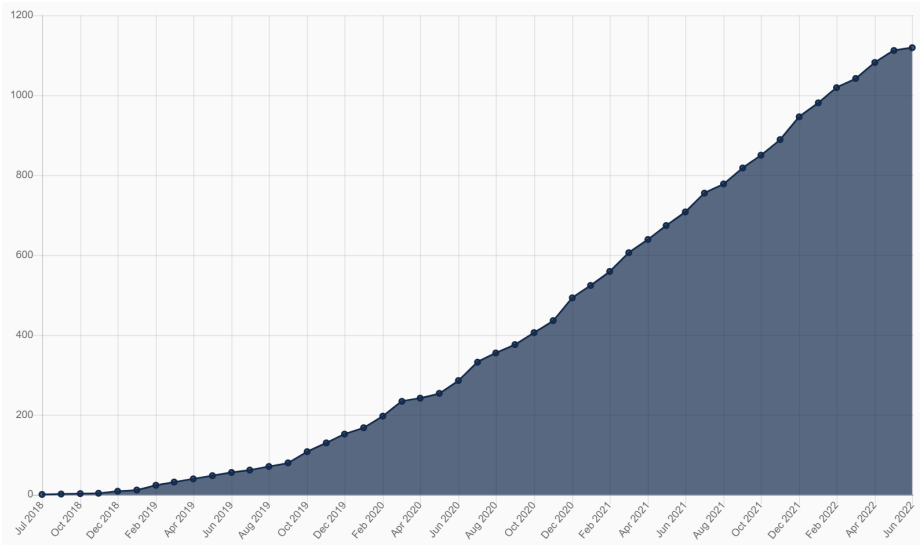
Neither the GDPR, nor Directive 95/46/EC clarify how such a de-identification process should or could be performed. Opinion 05/2014 of WP29 [48] classifies data anonymization techniques common in practice into three categories: pseudonymization, randomization and generalization.

3.2.1 Pseudonymization

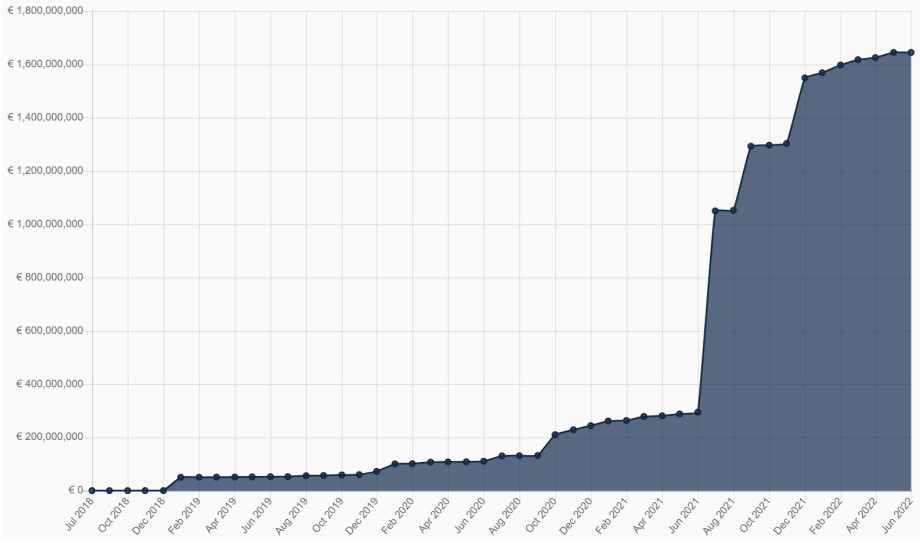
Pseudonymization is simply the replacement of one attribute (typically a unique identifier or other attribute with unique values) by another. Pseudonymization can help reduce linkability, but it cannot be relied on exclusively for anonymization. Pseudonymization methods rely on methods from encryption to assign the pseudonyms.

There are various methods to implement pseudonymization, namely:

Hash functions A hash function is a mathematical function that takes the value of a data attribute and transforms it into a *hash code*. Hash functions are useful for pseudonymization, as multiple occurrences of the same attribute value for several data points will be preserved. This is illustrated in Figure 3.3.



(a) Cumulative number of fines



(b) Cumulative overall sum of fines

Figure 3.2: GDPR fines since its implementation [49]

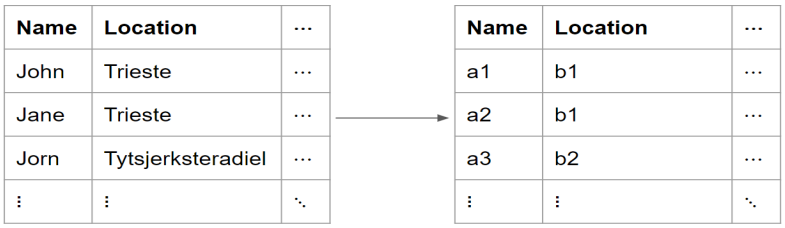


Figure 3.3: Hash functions applied to two personal attributes of a table. Note that Trieste is consistently mapped to the same hash code.

Cryptography Cryptographic techniques can also be used to transform attribute values into codes. Like hash functions, these methods preserve multiple occurences of the same value across records. Unlike hash functions, however, cryptographic methods are *invertible*. That is, if a user has the *key*, then (s)he can be transform codes back into the corresponding original names. Public-key cryptographic methods (e.g. RSA) may be useful in the context of pseudonymization, if multiple trusted parties need access to the real attribute values. A public, pseudomyzed version of the data can then be shared and only parties with the key can access the un-pseudomyzed attrbute values. This is illustrated in figure 3.4. By making the transformation invertible, cryptography imposes considerably more privacy risk than hash functions, which are not invertible.

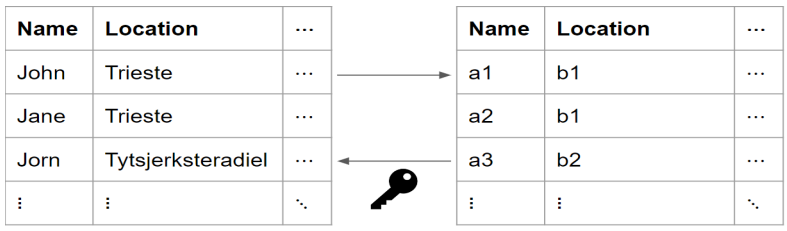


Figure 3.4: Pseudonymization through cryptography. The transformation can be inverted by anyone with access to the key.

Data obtained after applying pseudonymization cannot legally be considered fully anonymized [48, 55]. Individuals can easily be re-identified using values of other attributes. These values also make linkage attacks highly applicable. Pseudonymization methods are therefore insufficient as stand-alone privacy solution. Indeed, this is easily illustrated in computational experiments [3].

3.2.2 Randomization

Randomization is the introduction of statistical inaccuracies to the data. Parties accessing the data after randomization do not know true attribute values. However, if the inaccuracies are sufficiently small, the data may still be statistically representative of the underlying population. Several methods for randomization exist, including:

Noise addition Noise addition is a collection of methods that alter attribute values. Rather than containing literal attribute values, values of attributes after treatment are their original value plus or minus some error term ("ε"). This is illustrated in Figure 3.5. When applying noise addition, it is important to alter values enough to obscure personal information. At the same time, such alterations should not remove the overall distribution of the dataset, as this renders the anonymized data useless for statistical analysis.

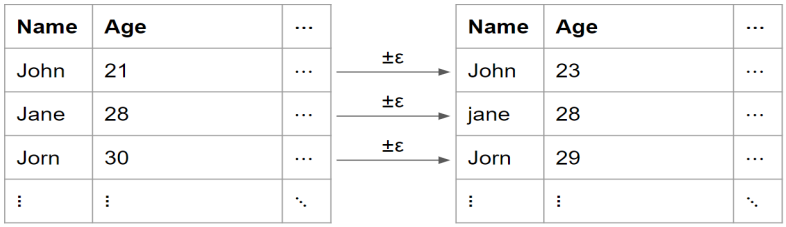


Figure 3.5: Randomization: noise addition

Permutation Permutation is the shuffling of a number of attribute values. After permutation, the values of (some of the) attributes no longer belong to the corresponding individual.

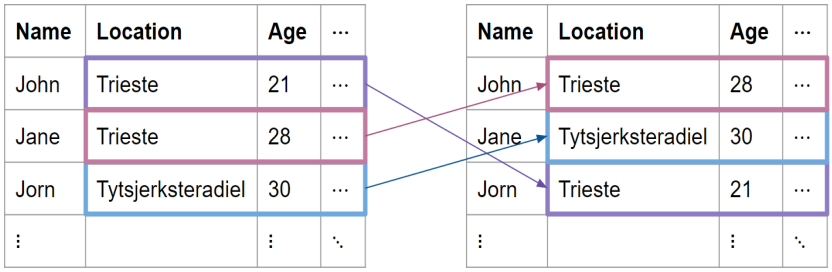


Figure 3.6: Randomization: permutation

Data anonymized through randomization was shown susceptible to reidentification and linkage attacks in practical cases [3, 44, 48, 54, 55, 56]. It offers no

protection against parameter inference attacks and is insufficient as a standalone anonymization method [48].

3.2.3 Generalization

Generalization is applied when specific values in a dataset are replaced by more general ones. This reduces the risk of individuals with highly specific attribute values being re-identified. For instance, in certain datasets, an individual's place of birth may be so unique that it is easy to infer the individual from it. Using country of birth (a more general category) instead reduces the risk of a re-identification attack, as more individuals share this attribute value. This is illustrated in Figure 3.7

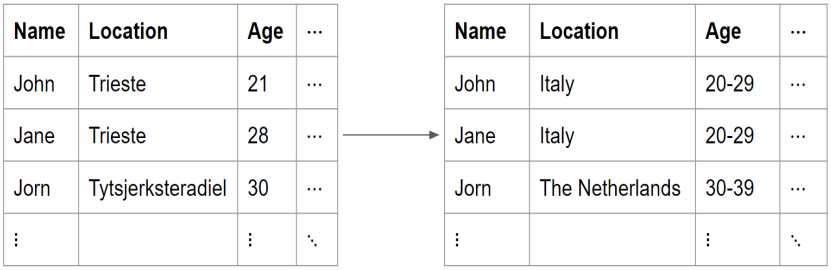


Figure 3.7: Generalization

The most common generalization framework is k-anonymity, in which attributes are generalized to the point where at least k individuals share each value. For example, suppose an attacker knows a data subject's city of birth, but this is generalized to a country of birth. Under k-anonymity, the attacker cannot identify the subject's record, as the country of birth is shared with at least $k-1$ other data subjects.

A further generalization step is l-diversity. This mechanism imposes the constraint that in each of the classes with at least k values, the remaining attributes take on at least l distinct values. l-diversity entails that not just individual attribute values, but also combinations of multiple attribute values cannot single out records. Consider again the example in which the location is generalized from city to country of birth. Under k-anonymity, there at least k individuals for any country of birth. Under l-diversity, there additionally are at least l records that have highly distinctive attribute values for each country of birth.

Generalization techniques significantly reduce the expressiveness of the information contained in a dataset. In 2016, a lawsuit forced the California Supreme Court to assess anonymization techniques. Of the inspected frameworks (pseudonymization, randomization, generalization), only k-anonymity was deemed sufficient in protecting privacy. Unfortunately, k-anonymity was proven to destroy the data's utility in the same lawsuit [55]. That is: while k-anonymity was the only traditional anonymization method to guarantee the

required degree of privacy, applying it meant the data became too generic for its intended use.

3.2.4 Differential Privacy

The aforementioned anonymization frameworks were all constructive in nature: they are classes of specific methods that alter the data with the intention of safeguarding the involved privacy. Generalization, however, had specific conditions under which data could be considered sufficiently anonymized (k-anonymity, l-diversity).

Diferential privacy (see, e.g. [57]) is a more recent paradigm in privacy enhancement. In this paradigm, data managers do not have direct access to data, but can query it through a system that returns aggregate results (descriptive statistics about attributes and combinations thereof). Differential privacy is a property of such a system: a system is considered *differentially private* if the manager cannot infer whether a given data subject is included in the dataset from any (combination of) query results.

To achieve the differential privacy property, systems typically also distort data prior to answering queries. This distortion is similar to those described in the other anonymization techniques. However, the data is distorted in an online fashion once the query is submitted. This means that the system's output differs even if the same query is submitted twice.

Example 3.2.1 *A data manager has a differentially private data system. (S)he wants to know the median income of data scientists in Trieste. As (s)he has no direct data access, (s)he cannot browse the records that satisfy these conditions. Instead, (s)he queries the system, instructing it to return the descriptive statistic of interest.*

By bypassing the need for direct data access, data systems significantly reduce the risks of both re-identification and linkage attacks [57]. However, if an attacker has detailed information about a data subject, (s)he can ask very specific questions that, to preserve differential privacy, the system cannot accurately answer. Multiple queries can also be combined to obtain specific sensitive information ("how many people live in Tytsjerksteradiel?" and "How many people not named Sebastiano live in Tytsjerksteradiel?" provide an answer to whether or not Sebastiano lives in said location). Arguably, differentially private systems may increase the risks of parameter inference attacks. As results are distorted in a different manner in every query, they may ultimately leak more information about a parameter's true value than a statically anonymized dataset.

Differential privacy offers improved protection over the previously outlined anonymization methods by restricting data access. This, however, directly impacts the data's utility. While the data manager can still obtain aggregate statistics, artificial intelligence techniques require direct data access. The anonymization techniques significantly reduced data utility through distortion, making the data unrepresentative of the true subject population. Differential privacy additionally directly limits the number of use cases to which the data can be

applied. For instance, in the case study in Section 2.2, access to the full dataset was needed to train the AI models. While differential privacy can help with elementary business intelligence, it is not suitable for training AI models, limiting innovation and forcing domain practitioners to rely on legacy technologies.

3.2.5 Synthetic Data Generation

Synthetic data generated using deep generative models form a new framework in secure data transfer. Artificial intelligence methods extract the patterns from the data that are analytically relevant. New, synthetic records are then generated, such that a dataset of synthetic records exhibits these same patterns. Every single datum is artificially generated, bearing no relation to entities in the original database. Synthetic data generation removes any link between available data and the individual. When using artificial data, the singularity of some records (e.g. a very specific location data) cannot be easily attributed to a single data subject. Due to the novelty of synthetic data, jurisprudence and established legal frameworks are unavailable.

A minor risk with generative models is that they may "overfit", learning and replicating exact patterns rather than randomly sampled ones. This can in theory lead to successful re-identification and linkage attacks even with synthetic data. However, this risk is only significant in small datasets and mechanisms are available to avoid this from occurring (a trivial verification after constructing the dataset can suffice, more theoretical methods are also possible; these are detailed in Chapter 4). In consequence, synthetic data aptly fits the legal moniker of anonymous data.

Another risk specific to synthetic data is that of the *membership inference attack* [58, 59]. This is a specific subcategory of re-identification attacks in which the attacker has access to a synthetic dataset and wants to assess whether a given synthetic record is also a record of the original dataset. Such attacks can be successful against partially synthetic data. However, Zhang et al. [59] show that fully synthetic electronic health records can be deemed sufficiently protected against membership inference attacks, regardless of the synthetic data generation methodology. This is corroborated by Bellovin et al. [60], who state that even if a synthetic dataset reproduces a datum that corresponds to a real data point, it cannot be automatically identified as such.

Carlini et al. [61] consider the specific case of text generation, in which the synthetic texts may contain real secrets (inferred from real texts). Given that the attacker knows that the text contains a secret, they inspect the likeliness of him/her finding it. The researchers experimentally conclude that identifying the secret is possible only three out of seven times. Note that the text format imposes considerably more structure (grammar, narrative, etc.) on the data than is present in tabular data. Thus, the effectiveness rate of the attacks is an overestimation of that of synthetic tabular data. Furthermore, explicit conditions under which finding the secret was possible could not be formulated, making secret discovery a matter of chance [61].

Synthetic data can allow for a more open flow of information than previ-

ously possible, in line with the "Towards A Common European Data Spaces" initiative. For this reason, synthetic data generation is receiving increasing attention in legal discourse within EU institutions. For example, the European Data Protection Supervisor (EDPS), dedicated a workshop to synthetic data recently [62]. The Artificial Intelligence Act makes explicit mention of synthetic data, recognizing it as an autonomous framework of techniques, separate from anonymization [63] (see Article 54, titled "Further processing of personal data for the development in the AI regulatory test space of certain AI systems in the public interest").

3.3 Conclusion

While the European Commission strives for increased data mobility in Europe, it struggles to reconcile this with the need to protect its citizens' personal information. Personal information is defined through a collection of attribute types. Date privacy is violated if a data subject can be re-identified directly, through linking multiple databases, or when information about a parameter is be inferred. Since the coming into force of the general data protection regulation, the number of violations has been steadily increasing.

Privacy enhancement methods can be classified into five categories. Of these, psuedonymization, randomization and generalization have the longest history. These three methods have been shown inadequate in countless data breaches. Moreover, they alter the data, limiting its analytic use. Resulting distorted data can only be considered secure if the degree of distortion fully outweighs the information contained in the dataset in the first place.

Differential privacy is an alternative in which the risks of re-identification and linkage are reduced. Differential privacy is a property of a data system, providing a formal mathematical definition of privacy. Unfortunately, this definition does not coincide with data privacy definitions in jurisprudence. Furthermore, differentially private systems only return aggregate data, dramatically limiting the number of use-cases by excluding methods from artificial intelligence. Furthermore, by applying distinct distortions whenever aggregate data is computed, differentially private systems may increase the risks of parameter inference attacks.

Synthetic data generation is an emerging technology in which privacy is provided by removing any link between available data and any real individual. Due to the novelty of the field, jurisprudence is lacking. However, the technology is hailed as a potential superior privacy protection method by scientists and policy makers.

Chapter 4

Synthetic Data Generation

In this Chapter, we address the research question *"Can variational autoencoder models be applied to generate disclosure averse and utility preserving tabular and relational data and if so, how can their effectiveness be measured?"*. In doing so, we first provide an intuitive explanation of the involved terminology, along with a motivation for the choice of variational autoencoders. We then formalize these notions to obtain a mathematically exact problem formulation. This allows us to define effectiveness metrics and develop suitable generative models for both tabular and relational datasets. We also introduce novel preprocessing techniques to handle advanced (combinations of) data types and provide an overview of how the methods efficiently can be implemented.

4.1 Motivation

In this chapter, we derive deep-learning methods for generation of realistic synthetic tabular and relational data. Two deep-learning frameworks for synthetic data generation have rose to prominence in the past few years. The first of these is that of *Generative Adversarial Networks* (GANs). Creswell et al. [64] and Aggarwal et al. [65] provide accessible introductions to GANs.

Intuitively, GANs consist of two interconnected neural networks: the *generator* and the *discriminator*. The generator is trained to produce synthetic data points, and the generator is trained to classify data points as either real or synthetic. The two networks are trained in a zero-sum game, in which the generator, by generating very realistic points, aims to maximize the number of false positives, whereas the discriminator aims to minimize this same quantity. With an origin in image generation, GANs are usually implemented as convolutional neural networks. They have for instance been shown useful in image segmentation [65].

The second framework is that of variational autoencoders (VAE). Doersch [66] provides an accessible introduction to VAE. Intuitively, VAE are based on the more striaght-forward autoencoders (AE). These are compression algorithms in which an *encoder* maps each data point to a *code*, a smaller representation of the data point. A *decoder* can then invert this process, inferring the original data point from the code. AE are deterministic algorithms, using one code for each point in a dataset. VAE are obtained when there there is no one deterministic code per datum. Instead, in VAE, the data are used to build one probabilistic representation of the overall distribution of the input dataset. Random samples can then be taken from this probability distribution, much like in traditional stochastic simulation (e.g. Monte Carlo simulation).

In this thesis, we chose VAE-based models for synthetic data generation. Training GANs involves starting with randomly generated data points from a latent space. While this works well for image processing, it is inefficient in the context of tabular data, where more structure is imposed by the multitude of possible data types. Tabular data also makes the results more dependent on the random initial conditions, hindering the robustness of the output for tabular and relational data. We have, however, also included introspective VAE models. These combine the strengths of both VAE and GAN-based methods by inferring patterns directly from the data (like VAE-based models), but also using a discriminator network for fine-tuning results (like GAN-based models). In particular, the inclusion of a discriminator helps identify relationships that are causal rather than merely correlations.

4.2 Relation to Prior Research

Most previous studies addressed the problem of synthetic data generation aiming at solving one of the two following problems: (a) the scarcity of real, actual data for training a ML system; (b) the need of not disclosing some information when

giving the real data away. In both cases, DL proved to be an effective tool.

Su et al. [56] show that a combination of real and artificial data resulted in an increase in performance in image processing. In 2019, a purely synthetic dataset was used to train a Convolutional Neural Network (CNN) for object detection purposes [67]. Park et al. [6] developed an algorithm based on Generative Adversarial Networks (GANs) with an additional classifier named table-GAN to generate synthetic data that can be used to train machine learning models. The resulting models are subsequently applicable to the original data set. In Chatterjee et al. [68], the authors combine GANs with transfer learning to augment datasets to better train ML algorithms for image classification. Castelli et al. [69] consider the problem of fault detection in wireless networks based on data obtained through monitoring: in order to mitigate the scarcity of data, the authors propose the use of a GAN (and a variant named Wasserstein GAN) to generate synthetic telecommunication data related to Wi-Fi signal quality. These studies indicate that findings obtained through artificial datasets are also representative of the original dataset. Oliveira et al. [70] use VAE-based models for the design of novel molecules, and the prediction of their properties.

Studies that coped with data synthesis with the motivation of non-disclosure deal more commonly with tabular data. Three recent articles have shown the potential to generate tabular data with one record per entry using generative models. Xu et al. [71] propose a conditional GAN named CTGAN where a conditional generator is used to synthesize eight different datasets. Xu and Veeramachaneni [72] employ both a GAN variation named Tabular GAN (TGAN) to address the same problem. Li et al. [58] experimentally show that synthetic data generated with GANs is sufficiently protected against attacks aiming to identify whether specific synthetic data points are also present in the real dataset. They show this for both images and on single-table tabular data. All three studies focused on tabular data containing continuous and categorical data-types. All three papers only focus on data with one record per entry, relying on GAN models. In our study, we show that methods based on VAE have an equally promising performance. Moreover, these models are adapted to generate the more complex, yet practically relevant class of relational datasets.

4.3 Definitions and Problem Statement

We consider the case where an organization (customer) is looking for a ML-based solution provided by another organization (provider). The customer aims at obtaining a preview of the solution from the provider based on its customer, problem-specific data. However, the customer does not want to give the data to the provider, nor to disclose the underlying information. For this reason, the customer wants to generate some synthetic data such that the preview of the solution based on the simulated data estimates the quality of the solution that would be based on the actual problem-specific data.

In the next sections, we define this scenario in detail, introducing formal definitions for the key concepts and formally stating the problem to be solved.

4.3.1 Data

A *dataset* D is a collection of one or more tables $\{T_1, T_2, \dots\}$. A *table* T is a collection of rows $\{t_1, \dots, t_n\}$. A row t is a tuple defined over a sequence of attributes $A(T) = (a_1, \dots, a_p)$ and represents an *entity* in the real world: we denote by $v(t, a) \in V(T, a) \cup \{\varnothing\}$ the value of the attribute a for the row t, with $V(T, a)$ being the *domain* of the attribute a and $\varnothing$ representing the undefined value (i.e., $v(t, a) = \varnothing$ means that a is not defined for t). We say that an attribute a is *unique* for a table T if and only if (a) $\forall t_i, t_j \in T, t_i \neq t_j \Rightarrow v(t_i, a) \neq v(t_j, a)$ and (b) a is always defined in T.

We say that a dataset D is *relational* if the following conditions hold: (a) it contains at least two tables; (b) at least one *primary table* $T^\star$ has a unique attribute $a^\star$; (c) for each other *secondary* table T in the dataset, $A(T) \ni a^\star$ and $\forall t \in T, \exists t^\star \in T^\star : v(t, a^\star) = v(t^\star, a^\star)$. Intuitively, a relational dataset is a dataset with a primary table $T^\star$ describing some entities, one row per entity, and other tables describing some other entities, each one linked to one specific entity of $T^\star$. In this work, we deal with two kinds of datasets: those containing one table, that we call *single-table* datasets, and the relational ones.

The *type* of an attribute a determines the nature of its domain $V(T, a)$. We consider five cases:

- a *real-valued* attribute has a domain $V(T, a) \subseteq \mathbb{R}$;
- a *discrete* attribute has a domain $V(T, a) \subseteq \mathbb{Z}$;
- a *time* attribute has a domain $V(T, a) \subseteq \mathbb{N}$ and its values represent time instants;
- a *categorical* attribute has a finite domain $V(T, a)$ consisting of non-numerical items and without a natural ordering;
- a *binary* attribute has a domain $V(T, a)$ with only two values.

Without loss of generality, we assume that, in relational datasets, the unique attribute $a^\star$ of the primary table $T^\star$ is of type *discrete* and its values are $1, 2, \dots \in \mathbb{N}$, i.e., for each $t_i \in T^\star$, $v(t_i, a^\star) = i$.

We say that a dataset D' is *compatible* with a dataset D if the following conditions hold: (a) D and D' contain the same number of tables; (b) there exists a one-to-one mapping $\phi : D \to D'$ between tables of D and tables in D' such that a table T and its image $T' = \phi(T)$ have the same attributes and the attributes have the same domains, i.e., $A(T) = A(T')$ and $\forall a \in A(T), V(T, a) = V(T', a)$. Clearly, every dataset D is compatible with itself.

As an example, consider a relational dataset containing a table $T^\star_{\texttt{user}}$ describing users, with the $|A(T^\star_{\texttt{user}})| = 4$ attributes `id`*, `age`, `country`, `gender`, `id`* being unique, and a table $T_{\texttt{purchases}}$ describing purchases made by users, with the $|A(T_{\texttt{purchases}})| = 5$ attributes `id`*, `prodId`, `quantity`, `date`, `prodPrice`. The attributes `id`*, `prodId`, `gender`, and `country` would be categorical; `age`

and `quantity` would be discrete; `prodPrice` would be real-valued (likely, with $V(\texttt{prodPrice}, T_{\texttt{purchases}}) = \mathbb{R}^+$).

4.3.2 Problem Statement

The goal of this study is to propose a method that, given an *original* dataset D, produces a compatible *synthetic* dataset $\hat{D}$ such that:

(a) for every problem based on D, every solution of that problem, and every effectiveness metric, the effectiveness of the solution built on $\hat{D}$ well estimates the effectiveness of the solution build on D;

(b) does not disclose any information about the real-world entities described by D.

In other words, $\hat{D}$ should be *utility-preserving* with respect to D, because it should allow to predict the utility of any solution in solving any problem based on D—in particular, it should allow to tell apart good solutions, i.e., those with great effectiveness, and bad solutions. And $\hat{D}$ should be *disclosure-averse* with respect to D, because it should not reveal information about entities described in D.

Because (i) the set of combinations of problem, solution, effectiveness metric is potentially infinite and (ii) measuring the degree to which the information is disclosed is hard, strictly verifying the two conditions above in practice is very hard. To overcome this limitation, in this chapter we introduce a few ways for measuring *utility-preserving* and *disclosure-averse* abilities of a synthetic dataset. We present them in the following sections.

4.3.3 Synthetic Data Performance Metrics

We propose three ways for measuring the utility-preservation (UP) of a synthetic dataset $\hat{D}$ with respect to a dataset D. One of them is *extrinsic*, i.e., it measures UP considering the effects of using $\hat{D}$ after using it for building a solution to a problem. The other two are *intrinsic*, i.e., they do not use $\hat{D}$ for actually building solutions and instead take into account some properties of $\hat{D}$ and D.

Model compatibility (MC) The model compatibility simply instantiates the idea that the greater the UP, the closer the effectiveness of a model $\hat{m}$ built on $\hat{D}$ to the effectiveness of a model m build on D, when both are assessed on D. As such, MC is an extrinsic measure of UP.

More formally, given a problem, a solution, and an effectiveness metric e, we define the MC of $\hat{D}$ with respect to D as:

$$\mathrm{MC}\left(\hat{D}, D\right) = \left|1 - \frac{e\left(m, D_{\text{test}}\right)}{e\left(\hat{m}, D_{\text{test}}\right)}\right| \tag{4.1}$$

where D_{test} is a portion of D compatible with D, m is a model learned with the solution on the remaining portion D_{train} of D, and $\hat{m}$ is a model learned with the solution on $\hat{D}$.

In practice, $\text{MC}(\hat{D}, D)$ considers the ratio between the effectiveness of the model learned on (a portion of) the original dataset and assessed on (a different portion of) the original dataset and the effectiveness of a model learned on the synthetic dataset $\hat{D}$ and assessed on (a portion of) the original dataset. The closer the two effectiveness values, the closer $\text{MC}(\hat{D}, D)$ to 0. That is, the more similar $\hat{D}$ and D.

Pairwise correlation difference (PCD) The pairwise correlation difference captures the idea that the correlation between pairs of attributes in the synthetic and original dataset should be similar. Measuring PCD does not require applying a solution for learning a model: PCD is hence an intrinsic measure of UP.

More formally, given the dataset D and $\hat{D}$ and the mapping ϕ between their tables, we define the PCD of $\hat{D}$ and D as:

$$\text{PCD}\left(\hat{D}, D\right) = \frac{1}{|D|} \sum_{T \in D} \frac{1}{\alpha_T} \left\| \boldsymbol{C}(T) - \boldsymbol{C}(\phi(T)) \right\|_F \tag{4.2}$$

where $|D|$ is the number of tables in D (and hence $\hat{D}$) and $\boldsymbol{C}(T)$ is the correlation matrix of attributes in T, α_T is a normalization factor, and $\|\cdot\|_F$ is the Forbenius norm of matrices, i.e., $\|\boldsymbol{A}\|_F = \sqrt{\sum_i \sum_j |a_{i,j}|^2}$. We assume that a proper measure of correlation is available for any possible pair of attribute types. Concerning the normalization factor, we set it to $\alpha_T = \sqrt{4\left(|A(T)|^2 - |A(T)|\right)}$, with $|A(T)|$ being the number of attributes in the table and hence the size of the matrix, in such a way that PCD takes values in $[0, 1]$.

In practice, $\text{PCD}\left(\hat{D}, D\right)$ considers the correlation matrices of all the tables in the datasets, measures the difference, and averages them. The more similar the correlation matrices, the closer $\text{PCD}\left(\hat{D}, D\right)$ to 0. That is, the more similar $\hat{D}$ and D.

Cluster synthetic evenness (CSE) The cluster synthetic evenness is applicable only to single-table datasets and investigates the tendency of the synthetic data to groups in clusters similarly to the original data. Similarly to PCD, measuring CSE does not involve learning a model; thus, CSE is an intrinsic measure of UP.

More formally, given the two single-table datasets D and $\hat{D}$, consisting respectively of the tables T and $\hat{T}$, and a clustering technique, we define the CSE for D and $\hat{D}$ as:

$$\text{CSE}\left(\hat{D}, D\right) = \frac{\alpha}{k} \sum_{i=1}^{i=k} \left| \frac{\left|C_i \cap \hat{T}\right|}{|C_i|} - \frac{\left|\hat{T}\right|}{|T| + \left|\hat{T}\right|} \right| \tag{4.3}$$

where C_i is the i-th cluster among the k clusters obtained by applying the clustering technique to the union of T and $\hat{T}$ and α is a normalization factor, depending only on $|T|$ and $|\hat{T}|$, that makes CSE take on values in $[0, 1]$. Precisely, we set the value of α to:

$$\alpha = \begin{cases} \frac{|T|+|\hat{T}|}{|T|} & \text{if } |\hat{T}| \leq |T| \\ \frac{|T|+|\hat{T}|}{|\hat{T}|} & \text{otherwise} \end{cases} \tag{4.4}$$

In practice, CSE merges the original and synthetic rows (i.e., data points) and cluster them in k clusters. Then, it measures the proportion of synthetic data in each cluster and compares this figure against the overall proportion of synthetic data. Finally, it averages the difference of the proportions across all clusters. The smaller the average difference, the more evenly the synthetic data distributes across clusters, i.e., the more similar are synthetic and original data in the way they group together. Practical experiments that indicate how well the metric measures the intended property are provided in Appendix A.1.

Disclosure-averseness (DA)

For measuring the degree to which a synthetic dataset $\hat{D}$ prevents disclosure of the information about entities described in the original dataset D, i.e., the disclosure-averseness (DA) of $\hat{D}$, we propose three intrinsic measures, described below.

Differently than for UP, we do not propose any extrinsic measure, i.e., a measure that aims at capturing the amount of information that is disclosable upon a reasonably meaningful attempt of disclosure. Despite, in principle, such a measure would be desirable, it is in practice very hard to devise it, since its soundness would greatly depend on the concrete nature of information, its value, and on a realistic modeling of the disclosure attempt, i.e., on a realistic threat model. Very likely, meeting all these requirements would make the measure very specific.

Nearest neighbor distance ratio (NNDR_μ, NNDR_σ, and NNDR_p)
The three intrinsic measures of DA are all based on the the concept of nearest neighbor and are applicable only for single-table datasets and with a proper distance function d between rows of the table T (and $\hat{T}$, that has the same attributes). More formally, given the two single-table datasets D and $\hat{D}$, consisting respectively of the tables T and $\hat{T}$, we define the mean nearest neighbor distance ratio for D and $\hat{D}$ as:

$$\begin{aligned} \text{NNDR}_\mu\left(\hat{D}, D\right) &= \left| 1 - \frac{1}{|T|} \sum_{t \in T} \frac{\min_{\hat{t} \in \hat{T}} d\left(t, \hat{t}\right)}{\min_{t' \in T \setminus \{t\}} d\left(t, t'\right)} \right| \\ &= \left| 1 - \frac{1}{|T|} \sum_{t \in T} \rho(t) \right| \end{aligned} \tag{4.5}$$

In practice, NNDR_μ considers each original row $t \in T$ and measures its distances to the closest original and synthetic rows: if the ratio $\rho(t)$ between these figures satisfies $\rho(t) > 1$, then t is closer to an original, than a synthetic row; else, if $\rho(t) < 1$, then t is closer to a synthetic row. Then, NNDR_μ averages the ratio across all original rows and tells its distance from 1: the closer the average to 0, the smaller the difference between closest distances to synthetic and original data. That is, the harder the task of re-identifying a real entity from synthetic data.

Similarly, we define the standard deviation of NNDR as:

$$\text{NNDR}_\sigma\left(\hat{D}, D\right) = \sqrt{\frac{\sum_{t\in T}\left(\rho(t) - \frac{1}{|T|}\sum_{t\in T}\rho(t)\right)^2}{|T|-1}} \tag{4.6}$$

In practice, the lower NNDR_σ, the more even the value of ρ across entities, i.e., the harder to identify some of them (for the same value of NNDR_μ). Moreover, if NNDR_σ is large and NNDR_μ is close to 1, it follows that a set of synthetic rows may be too close to original rows, and the remaining are very far.

In order to capture the quantity of original rows that are too close to synthetic rows, i.e., that could be identified easily, we define a further index:

$$\text{NNDR}_p\left(\hat{D}, D\right) = \left|\frac{1}{2} - \frac{1}{|T|}\left|\{t \in T : \rho(t) < 1\}\right|\right| \tag{4.7}$$

In practice, NNDR_p measures the rate of original rows that are closer to synthetic rows than to other original rows and then tells how this rate is close to the ideal value of 50 %: the lower NNDR_p, the lower the proportion of original rows that are closer to synthetic rows than to other original rows.

Nearest neighbor distributions difference (NNDD) Finally, we define the nearest neighbor distributions difference as follows. Let X and $\hat{X}$ the distributions of the values of the distance to the closest original and synthetic neighbors across original rows $t \in T$, then $\text{NNDD}\left(\hat{D}, D\right)$ is 1 if the null-hypothesis that the two distributions are different cannot be rejected and 0 otherwise. In practice, if the distributions are statistically the same, it would be hard to find a pattern in distances among real and synthetic entities that is exploitable for telling apart the former from the latter. In our experiments, we perform the statistical significance test for the null-hypothesis using the Kolmogorov-Smirnov test and a predefined value for α.

4.4 Deep-Learning Generative Models

4.4.1 Variational Autoencoders

We considered three variants of VAEs for generating numerical data: standard variational autoencoders (VAE), beta-variational autoencoders (β-VAE), and

introspective variational autoencoders (introVAE). These methods were originally conceived in the field of image processing.

VAEs form a paradigm in generative machine learning, in which an algorithm referred to as the *encoder* infers a probabilistic latent representation in $\mathbb{R}^r$ (the latents space, also named *code*) of an input numerical dataset in $\mathbb{R}^q$. A *decoder* is an algorithm trained to take random samples from the latent representation and convert them back to $\mathbb{R}^q$. Both the encoder and decoder are typically artificial neural networks and they are learned concurrently using a combined loss function [73]. The dimension r of the latent space, as well as the architecture (i.e., number of hidden layers, activation function, etc.) of the encoder and the decoder are hyperparameters.

β-VAEs form an extension of the VAE modeling framework. This methodology was originally developed to infer disentangled latent representations. In such representations, each latent variable affects observable attributes in an individual manner, rather than having single latent variables encoding complex interactions of factors. A disentangled representation enables users to control individual factors of variation. Disentanglement is achieved through requiring the variables in the latent space to be statistically independent. β-VAEs are promising in syntethic data generation as they allow to explore different trade-offs between realism of synthetic data and the strictness of this independence constraint. In practice, both objectives are terms in the loss function, and an additional hyperparameter β is used to manage their relative importance [4].

IntroVAEs form a hybrid framework that combine VAE with generative adverserial networks (GANs) [74]. In GANs, two neural networks are trained concurrently: a generator that samples synthetic data points from a latent spaces and a discriminator, trained to make distinctions between empirical and synthetic data points. By training both networks with one loss function, a zero-sum game is obtained. This is because the generator aims to maximize the number of times the discriminator incorrectly classifies an artificial data point. Naturally, the discriminator aims to minimize this quantity. In IntroVAE, a standard VAE model is augmented with a discriminator. The VAE generates artificial data points, and the discriminator classifies data points as either real or synthetic. The incorporation of the discriminator results in synthetic data in which realism is achieved on a finer scale [5]. While VAE and β-VAE guarantee that coarse patterns in data are preserved, IntroVAE enable the generation of synthetic data that is realistic at the level of individual data points. This is because the discriminator infers and exploits particular inter-dependencies between attributes that (β-)VAE loss functions, typically based on Kullback-Leibler (KL) divergence cannot identify. In IntroVAE, two parameters for leveraging the trade-off between KL-divergence and reconstruction error are additional hyperparameters. Compared to GANs, the use of VAE as a generating mechanism results in accelerated convergence. IntroVAE are also less likely to overfit, reproducing exact patterns inferred from empirical data. Unfortunately, the discriminator in IntroVAE complicate the back-propagation algorithm for non-numeric data types. Therefore, we use the Gumbel softmax function as the activation function, with its temperature being a hyperparameter as outlined for GANs in [74].

The use of this loss function to make IntroVAE applicable to tabular data is a novel contribution of our work.

4.4.2 Preprocessing Techniques for Tabular and Relational Data Types

In general, we convert values of attributes whose type is one of the five types described in Section 4.3.1 to numerical vectors of a suitable size—that is, one value corresponds to a vector in $\mathbb{R}^p$ with $p \geq 1$. Formally, we define the conversion to and from numbers of values for the attribute a of a table T as a pair of functions $\phi_{\text{type}} : V(T, a) \to \mathbb{R}^p$ and $\phi^{-1}_{\text{type}} : \mathbb{R}^p \to V(T, a)$. We describe how we build ϕ_{type} and ϕ^{-1}_{type}, given a table $T \in D$ and an attribute $a \in A(T)$, for the five types in the following sections.

Eventually, we convert each table T in D to a collection $X = \{\boldsymbol{x}_i\}_i$ of numerical vectors, one vector per row, resulting from the concatenation of the single vectors obtained by converting each attribute:

$$\begin{aligned} \boldsymbol{x}_i &= [\boldsymbol{x}_{i,a_1}\ \boldsymbol{x}_{i,a_2}\ \dots] \\ &= [\phi_{\text{type}}(v(t_i, a_1))\ \phi_{\text{type}}(v(t_i, a_1))\ \dots] \end{aligned} \tag{4.8}$$

Consistently, when converting a collection of numerical vectors back to a table of rows with proper attributes (i.e., the same of T), we first split a vector $\boldsymbol{x}_i$ in chunks of proper size, then use the corresponding ϕ^{-1}_{type} to map each chunk to an attribute value of the row t_i. We denote these two steps respectively as $X := \phi_{\text{type}}(T)$ and $T := \phi^{-1}_{\text{type}}(X)$.

Real-valued and discrete attributes

For real-valued and discrete attributes, we use a quantile-based conversion. We remark that we convert also this kind of attributes, that are already numerical in the original dataset, in order to make them more VAE-friendly. Indeed, in image processing applications, where VAEs are particularly common, continuous numerical attributes typically represent color, hue, or saturation values, with magnitudes represented on the same scale. In tabular data, no such scale-indifference between numerical attributes can be guaranteed: for instance, in financial data, investments may be expressed in magnitudes of tens of thousands, whereas age is measured on a much smaller scale. This may make the VAE less effective, as attributes with larger values may be incorrectly be interpreted as having more impact. A quantile-based conversion is one of the method to cope with this problem [75].

Formally, let $\{v(t_i, a)\}_i$ be the defined values (i.e., not $\varnothing$) of the attribute a in the table T, with $\forall i, v(t_i, a) \in V(T, a) \subseteq \mathbb{R}$, since the attribute is numeric. We first compute the k quantiles $(q_1, \dots, q_k)$ of $\{v(t_i, a)\}_i$, with $q_1 = \min_i v(t_i, a)$ and $q_k = \max_i v(t_i, a)$ and $k \geq 2$ being a parameter of the conversion. Then,

we define $\phi_{\text{type}} : [q_1, q_k] \to [0, 1]$ as:

$$\phi_{\text{type}}(v) = \begin{cases} \frac{j}{k} & \text{if } v \neq \varnothing \\ 0.5 & \text{otherwise} \end{cases} \tag{4.9}$$

with j such that $v \in [q_j, q_{j+1}[$. We define $\phi^{-1}_{\text{type}} : [0, 1] \to [q_1, q_k]$ as:

$$\phi^{-1}_{\text{type}}(x) = q_j + (q_{j+1} - q_j)(kx - j) \tag{4.10}$$

with $j = \lfloor kx \rfloor$.

In practice, an input numerical value is mapped to the corresponding percentile (scaled to $[0, 1]$); in the other direction, a value in $[0, 1]$ is mapped to the corresponding quantile interval and, inside it, linearly scaled. As a result, all real-valued and discrete attributes are mapped to $[0, 1]$ (thus $p = 1$), hence mitigating the scale problem.

Time attributes

Time attributes typically represent timestamps of events related to an entity. When converting time attributes, we consider the time density of these events, we attempt to infer the periodic component of this density and decouple it from the trend component, and we map these components to normalized values.

Formally, let $\{v(t_i, a)\}_i$ be the defined values of the attribute a in the table T, with $\forall i, v(t_i, a) \in V(T, a) \subseteq \mathbb{N}$—in practice, time values are expressed using numbers, as in the Unix time format. First, we compute the earliest value $\tau_{\min} = \min_i v(t_i, a)$ and the latest value $\tau_{\max} = \max_i v(t_i, a)$. Second, we build a time series $\boldsymbol{b}$ with $\frac{1}{\delta\tau}(\tau_{\max} - \tau_{\min})$ elements, where $\delta\tau$ is a parameter of the conversion, as:

$$b_j = |\{i : \tau_{\min} + j\delta\tau \leq v(t_i, a) < \tau_{\min} + (j + 1)\delta\tau\}| \tag{4.11}$$

that is, b_j is the number of values within the j-th $\delta\tau$-long interval. Third, we compute the discrete cosine transform (DCT) of $\boldsymbol{b}$ and obtain the period $\tau^\star$ of the strongest harmonic of $\boldsymbol{b}$. Then, we define $\phi_{\text{type}} : [\tau_{\min}, \tau_{\max}] \to [0, 1]^2$ as:

$$\phi_{\text{type}}(v) = \begin{cases} (x_1, x_2) & \text{if } v \neq \varnothing \\ (0,5, 0.5) & \text{otherwise} \end{cases} \tag{4.12}$$

with:

$$x_1 = \left\lfloor \frac{v - \tau_{\min}}{\tau^\star} \right\rfloor \frac{\tau^\star}{\tau_{\max} - \tau_{\min}} \tag{4.13}$$

$$x_2 = \frac{v \bmod \tau^\star}{\tau^\star} \tag{4.14}$$

where mod means the remainder of the real division. We define $\phi^{-1}_{\text{type}} : [0, 1]^2 \to [\tau_{\min}, \tau_{\max}]$ as:

$$\phi^{-1}_{\text{type}}(\boldsymbol{x}) = \phi^{-1}_{\text{type}}(x_1, x_2) = \tau_{\min} + x_1 \left(\tau_{\max} - \tau_{\min}\right) + x_2\tau^\star \tag{4.15}$$

In practice, given the periodicity $\tau^\star$ computed considering the DCT, x_1 represents the period in which the value $v(t_i, a)$ falls in and x_2 represents the offset with respect to the starting time of that period.

Categorical and binary attributes

For categorical and binary attributes we rely on one-hot encoding.

Formally, let $c_1, \ldots, c_k$ be the different categorical values (with $k = 2$ for binary attributes) in $C \subseteq V(T, a)$. We define $\phi_{\text{type}} : C \to [0, 1]^{|C|}$ as:

$$\phi_{\text{type}}(v) = \begin{cases} \boldsymbol{x} & \text{if } v \neq \varnothing \\ (0.5, \ldots, 0.5) & \text{otherwise} \end{cases} \tag{4.16}$$

with $\boldsymbol{x} \in \{0, 1\}^k$ and:

$$x_i = \begin{cases} 1 & \text{if } v = c_i \\ 0 & \text{otherwise} \end{cases} \tag{4.17}$$

We define $\phi^{-1}_{\text{type}} : [0, 1]^{|C|} \to C$ as:

$$\phi^{-1}_{\text{type}}(\boldsymbol{x}) = c_j \tag{4.18}$$

with $j = \arg\max_i x_i$.

Missing values

In order to support datasets with missing values, that are rather common in business data, we perform a further processing for each attribute of each table for which there are missing values, i.e., for which at least one value is $\varnothing$.

Formally, for each attribute a and each table T such that for at least one row $t_i \in T$ the value $v(t_i, T) = \varnothing$, we introduce a further numerical value in the conversion (besides those deriving from the type conversions described above) that encodes the fact that the value is undefined (i.e., $\varnothing$). We define this further mapping based on the previously described cases:

$$\phi'_{\text{type}}(v) = \begin{cases} [\phi_{\text{type}}(v)\ 0] & \text{if } v \neq \varnothing \\ [\phi_{\text{type}}(v)\ 1] & \text{otherwise} \end{cases} \tag{4.19}$$

and:

$$\phi'^{-1}_{\text{type}}(\boldsymbol{x}) = \begin{cases} \phi^{-1}_{\text{type}}(v) & \text{if } x_{\text{last}} > 0.5 \\ \varnothing & \text{otherwise} \end{cases} \tag{4.20}$$

where x_{last} is the last element of $\boldsymbol{x}$.

In practice, for attributes that can be undefined, we append one further element to the vector output by the conversion that is 1 if the value is missing

and 0 otherwise. When converting back to the attribute type, we use the last element to decide if the value has to be set to $\varnothing$.

Summarizing, each value of an attribute is converted to a numerical vector in $[0,1]^p$, with $p = 1$ for real-valued and discrete attributes, p is the number $|C|$ of distinct categorical values for categorical and binary attributes, and $p = 2$ for time attributes. When attributes have missing value, p is increased by 1.

Having described how to convert tables in collection of numerical vectors (and the opposite conversion), the problem of synthetic dataset generation can be split in five steps. For ease of presentation, we first describe those steps for the single-table dataset case, then we will refine the description for the more complex case of relational datasets.

Given a single-table dataset D with a table T, we:

1. transform T to $X = \{\boldsymbol{x}_i\}_i = \phi_{\text{type}}(T)$, with each $\boldsymbol{x}_i \in [0,1]^q$;
2. learn a VAE on X, i.e., a pair consisting of a encoder $\phi_{\text{enc}} : \mathbb{R}^q \to \mathbb{R}^r$ and a decoder $\phi_{\text{dec}} : \mathbb{R}^r \to \mathbb{R}^q$;
3. generate a collection $\hat{Y} = \{\hat{\boldsymbol{y}}_j\}_j$ of numerical vectors, with each $\hat{\boldsymbol{y}}_j \in \mathbb{R}^r$;
4. obtain a collection of $\hat{X} = \{\hat{\boldsymbol{x}}_j\}_j$ of numerical vectors, with each $\hat{\boldsymbol{x}}_j = \phi_{\text{dec}}(\hat{\boldsymbol{y}}) \in [0,1]^q$ by applying the decoder to each element in Y;
5. transform $\hat{X}$ to a synthetic table $\hat{T} = \phi^{-1}_{\text{type}}(\hat{X})$ by converting back to an attribute value each chunk of $\hat{\boldsymbol{x}}$.

Learning a VAE

From the point of view of our approach, all the three variants (standard VAE, β-VAE, and IntroVAE) can be seen as "black boxes" that given a dataset $X = \{\boldsymbol{x}_i\}_i$, with $\boldsymbol{x}_i \in \mathbb{R}^q$, a target latent space dimension r, and a suitable set of hyperparameters values (that is different among the three versions of VAE), output a pair of multivariate numerical functions $\phi_{\text{enc}} : \mathbb{R}^q \to \mathbb{R}^r$ and $\phi_{\text{dec}} : \mathbb{R}^r \to \mathbb{R}^q$ (encoder and decoder, respectively). The two functions are neural networks and are completely described by a vector $\boldsymbol{\theta} = [\boldsymbol{\theta}_{\text{enc}}\ \boldsymbol{\theta}_{\text{dec}}] \in \mathbb{R}^m$ of numerical parameters, whose size m depends on the variant. We obtain $\boldsymbol{\theta}$ from X in a learning process which is a gradient-based optimization driven by a loss function $L : \mathbb{R}^m \to \mathbb{R}$ such that the lower $L(\boldsymbol{\theta})$, the better the pair $\phi_{\text{enc}}, \phi_{\text{dec}}$ described by $\boldsymbol{\theta}$.

The loss function is based on the assumption that the decoder and the encoder act as conditional probability distributions on the proper spaces. The values of X are considered observations stemming from an underlying probability distribution over a latent space Z. The decoder ϕ_{dec} defines a probability distribution depending on parameters $\boldsymbol{\theta}_{\text{dec}}$, such that $p(\boldsymbol{x}|\boldsymbol{z}, \boldsymbol{\theta}_{\text{dec}})$ is the probability that $\boldsymbol{x}$ was observed, given that the latent variables were $\boldsymbol{z}$ and the model parameters were $\boldsymbol{\theta}_{\text{dec}}$. Symmetrically, the encoder ϕ_{enc} defines the distribution of the latent space for given observations parametrized by $\boldsymbol{\theta}_{\text{enc}}$. Thus,

$p(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{\theta}_{\text{enc}})$ is the probability that $\boldsymbol{z}$ was the latent vector, given that $\boldsymbol{x}$ was observed.

Specifically, for standard VAEs we use the Kullback-Leibler (KL)-divergence between observations X and the distribution of generated data the following loss function:

$$\begin{aligned} L(\boldsymbol{\theta}) &= \mathbb{E}_{\boldsymbol{z} \sim p(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{\theta}_{\text{enc}})} \log p\left(\boldsymbol{x}|\boldsymbol{z}, \boldsymbol{\theta}_{\text{dec}}\right) \\ &= \mathbb{E}_{\phi_{\text{enc}}} \log \phi_{\text{dec}} \end{aligned} \tag{4.21}$$

where the latter is a shorthand for the former.

For β-VAEs, a constraint is imposed that the encoder ϕ_{enc} must also be sufficiently similar to a given prior $P(\boldsymbol{z})$ (of the latent space). Typically, the multivariate normal distribution $\mathcal{N}(\mathbf{0}, \mathbf{1})$ is chosen as the prior, which enforces statistical independence on the latent space. KL-divergence ia also invoked to achieve this. The constraint is incorporated into the loss function through the KKT-conditions. Hence, β-VAEs we use the following loss function:

$$L(\boldsymbol{\theta}) = -\mathbb{E}_{\phi_{\text{enc}}} \log \phi_{\text{dec}} + \beta D_{\text{KL}}\left(\phi_{\text{enc}} \parallel \mathcal{N}\right) \tag{4.22}$$

where $D_{\text{KL}}(\phi_{\text{enc}} \parallel \mathcal{N}(\mathbf{0}, \mathbf{1}))$ denotes the KL-divergence between distributions $p(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{\theta}_{\text{enc}})$ and $\mathcal{N}(\mathbf{0}, \mathbf{1})$. The parameter β is a hyperparameter used for scaling the relative importance of reconstruction accuracy and similarity to the prior.

Finally, for IntroVAEs we use the loss function specified in [5], that we do not report here for brevity.

Concerning the optimizer, we use ADAM with standards parametrization.

Latent data generation and mapping

Given a target size n of the synthetic data $n = |\hat{X}|$, we generate the collection $\hat{Y} = \{\hat{\boldsymbol{y}}_j\}_j$ of n points in the latent space as follows.

First, we compute the image of the input data X in the latent space, i.e., the collection $Y = \{\boldsymbol{y}_i\}_i$, with $\boldsymbol{y}_i = \phi_{\text{enc}}(\boldsymbol{x}_i) \in \mathbb{R}^r$; we denote this step as $Y := \phi_{\text{enc}}(X)$. Then, we fit a multivariate normal distribution over Y and obtain its parameters $\boldsymbol{\mu} \in \mathbb{R}^r$ and $\boldsymbol{\Sigma} \in \mathbb{R}^{r \times r}$; we denote this step as $(\boldsymbol{\mu}, \boldsymbol{\Sigma}) := \mathcal{N}^{-1}(Y)$. Finally, we sample the multivariate normal distribution $\mathcal{N}\left(\boldsymbol{\mu}, \boldsymbol{\Sigma}\right)$ n times and obtain $\hat{Y}$; we denote this step as $\hat{Y} :\overset{n}{\sim} \mathcal{N}\left(\boldsymbol{\mu}, \boldsymbol{\Sigma}\right)$. As the last step, we obtain $\hat{X} := \{\hat{\boldsymbol{x}}_j\}_j$ by applying the decoder to $\hat{Y}$, i.e., $\forall j, \hat{\boldsymbol{x}}_j := \phi_{\text{dec}}(\hat{\boldsymbol{y}}_j)$; we denote this step as $\hat{X} = \phi_{\text{dec}}(\hat{Y})$.

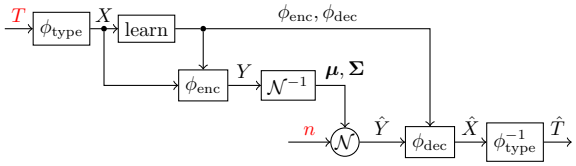


Figure 4.1: Schematic representation of the steps for going from a table T of a single-table dataset to a synthetic table $\hat{T}$. In red, the parameter and input provided by the user.

Summarizing, the entire process from T to $\hat{T}$ consists of the following steps:

$$X := \phi_{\text{type}}(T) \tag{4.23}$$
$$(\phi_{\text{enc}}, \phi_{\text{dec}}) := \text{learn}(X) \tag{4.24}$$
$$Y := \phi_{\text{enc}}(X) \tag{4.25}$$
$$(\boldsymbol{\mu}, \boldsymbol{\Sigma}) := \mathcal{N}^{-1}(Y) \tag{4.26}$$
$$\hat{Y} :\overset{n}{\sim} \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \tag{4.27}$$
$$\hat{X} := \phi_{\text{dec}}\left(\hat{Y}\right) \tag{4.28}$$
$$\hat{T} := \phi_{\text{type}}^{-1}\left(\hat{X}\right) \tag{4.29}$$

The same steps are depicted in Figure 4.1.

4.4.3 Relational datasets

Relational datasets require a different approach, since they are constituted by more than one table. We modified the approach described above in order to accommodate this difference and driven by a further two-fold goal: (1) generate synthetic secondary tables whose size are consistent with the corresponding original tables; (2) make several rows of the synthetic secondary tables refer to the same row in the synthetic primary table, as happens for the original data.

For achieving these goals, we use a multi-level VAE (ML-VAE), i.e., a single VAE variant that is different from the three variants used in the case of single-table datasets. ML-VAE were originally conceived for inferring disentangled latent representations when data is not independent, identically distributed [76]. The use of this methodology for generating synthetic dataset with more than one table is a novel contribution of our work. The original method proposed in [76] operates by stratifying the data into strata with shared characteristics: each of the characteristics upon which stratification is based are then mapped to one latent variable during encoding. A disentangled latent representation is then obtained in which the properties of each stratum correspond to a unique variable in the code. In our case, instead, we use the disentangled latent variables to represent the row in the primary table to which a point refers to.

In practice, we repeat a slightly modified version of the steps described in Section 4.4.2 one time for each table in the dataset and add, for each table, a pre-processing step and a post-processing step. Let the original dataset D be composed by the primary table $T^\star$ and by h secondary tables $T^1, \ldots, T^h$. We proceed as follows.

Primary table

We pre-process the primary table $T^\star$, obtaining a table $T'^\star$, by adding one new discrete attribute a_{T^l} for each secondary table T^l. For each row t in $T^\star$ and each l, we set the value $v\left(t, a_{T^l}\right)$ to the number of rows t' in T^l such that $v(t', a^\star) = v(t, a^\star)$. In other words, a_{T^l} for t represents the number of secondary entities, i.e., entities of a secondary table, that refer to the primary entity described by t. We also remove $a^\star$ from $A(T^\star)$.

Then, we learn a β-VAE $\phi^\star_{\text{enc}}, \phi^\star_{\text{dec}}$, with $r^\star$ as latent space dimension, on $X^\star = \phi_{\text{type}}(T'^\star)$.

We build $Y^\star$, $\hat{Y}^\star$, and $\hat{X}^\star$ in the same way of the single-table case: in particular, we use n as the user-provided size of the synthetic primary table to be generated from $\hat{X}^\star$. Then, when building the synthetic primary table, we apply ϕ^{-1}_{type} to $\hat{X}^\star$, obtaining $\hat{T}'^\star = \phi^{-1}_{\text{type}}(\hat{X}^\star)$, and finally post-process $\hat{T}'^\star$, obtaining $\hat{T}^\star$, by removing the attributes $a_{T^1}, \ldots, a_{T^h}$ and by adding again $a^\star$, whose values are set to $v(\hat{t}_j, a^\star) = j$.

Summarizing, for the primary table $T^\star$ we do the following steps in order to obtain the corresponding synthetic primary table $\hat{T}^\star$:

$$T'^\star := \text{preProc}\left(T^\star\right) \tag{4.30}$$
$$X^\star := \phi^\star_{\text{type}}\left(T'^\star\right) \tag{4.31}$$
$$\left(\phi^\star_{\text{enc}}, \phi^\star_{\text{dec}}\right) := \text{learn}\left(X^\star\right) \tag{4.32}$$
$$Y^\star := \phi^\star_{\text{enc}}\left(X^\star\right) \tag{4.33}$$
$$\left(\boldsymbol{\mu}^\star, \boldsymbol{\Sigma}^\star\right) := \mathcal{N}^{-1}\left(Y^\star\right) \tag{4.34}$$
$$\hat{Y}^\star :\overset{n}{\sim} \mathcal{N}\left(\boldsymbol{\mu}^\star, \boldsymbol{\Sigma}^\star\right) \tag{4.35}$$
$$\hat{X}^\star := \phi^\star_{\text{dec}}\left(\hat{Y}^\star\right) \tag{4.36}$$
$$\hat{T}'^\star := \phi^{-1}_{\text{type}}\left(\hat{X}^\star\right) \tag{4.37}$$
$$\hat{T}^\star := \text{postProc}\left(\hat{T}'^\star\right) \tag{4.38}$$

Secondary tables

We repeat the same procedure, described below, for each secondary table T^l.

First, similarly to the case of the primary table, we pre-process T^l by removing the attribute $a^\star$, hence obtaining T'^l. We remark that, since we never alter the ordering of items in the collections that we work with (each T^l and its

corresponding T'^l, X^l, and Y^l), we are still able to track items back to the row in the primary table they refer to.

Then, we learn a ML-VAE $\phi^l_{\text{enc}}, \phi^l_{\text{dec}}$ from X^l. In this ML-VAE, the encoder is a function $\phi^l_{\text{enc}} : \mathbb{R}^{q^l} \to \mathbb{R}^{r^l}$, q^l being the size of elements of X^l and r^l being the user-provided size of the latent space for this secondary table. Differently than previous cases, here the decoder is a function $\phi^l_{\text{dec}} : \mathbb{R}^{r^l + r^\star} \to \mathbb{R}^{q^l}$: that is, the dimension of the source space for the decoder is larger than the dimension of the destination space of the encoder. Precisely, the former has $r^\star$ more variables than the latter. When learning the ML-VAE on X^l, i.e., when computing the loss for elements $\{\boldsymbol{x}^l_i\}_i$ of X^l, each point in the latent space is obtained as a row-wise concatenation:

$$\boldsymbol{y}^l_i = [\overbrace{\phi^l_{\text{enc}}\left(\boldsymbol{x}^l_i\right)}^{\in \mathbb{R}^{r^l}} \overbrace{\phi^\star_{\text{enc}}\left(\boldsymbol{x}^\star\right)}^{\in \mathbb{R}^{r^\star}}] \in \mathbb{R}^{r^l + r^\star} \tag{4.39}$$

where $\boldsymbol{x}^\star$ is the element of $X^\star$ corresponding to the row $t^\star$ of the primary table such that $v(t^\star, a^\star) = v(t^l_i, a^\star)$, i.e., the primary row related to the row $\boldsymbol{x}^l_i$ of the secondary table T^l. As a shorthand, we write collectively that:

$$\begin{aligned} Y^l = \{\boldsymbol{y}^l_i\}_i &= \left\{[\phi^l_{\text{enc}}\left(\boldsymbol{x}^l_i\right) \;\; \phi^\star_{\text{enc}}\left(\boldsymbol{x}^\star\right)]\right\}_i \\ &= \left[\phi^l_{\text{enc}}\left(X^l\right) \phi^\star_{\text{enc}}\left(\text{prim}\left(X^l\right)\right)\right] \end{aligned} \tag{4.40}$$

When learning this ML-VAE, we use the same optimizer described in Section 4.4.2 and the following loss function:

$$L(\boldsymbol{\theta}) = \frac{1}{|T^\star|} \sum_{l=1}^{|D|-1} \sum_{v^\star \in V(T^\star, a^\star)} (L_{\text{reg}}(l, v^\star, \boldsymbol{\theta}) - \beta_l L_{\text{KL}}(l, v^\star, \boldsymbol{\theta})) \tag{4.41}$$

where:

$$L_{\text{reg}}(l, v^\star, \boldsymbol{\theta}) = \sum_{t^l_i : v(t^l_i, a^\star) = v^\star} \mathbb{E}_{\phi^l_{\text{enc}}(v^\star)} \mathbb{E}_{\phi^l_{\text{enc}}(t^l_i)} \log \phi^l_{\text{dec}} \tag{4.42}$$

$$L_{\text{KL}}(l, v^\star, \boldsymbol{\theta}) = \sum_{t^l_i : v(t^l_i, a^\star) = v^\star} D_{\text{KL}}(\phi^l_{\text{enc}}(t^l_i) \parallel P_{(l, v^\star)}) \tag{4.43}$$

and where $\phi^l_{\text{enc}}(v^\star)$ denotes the encoder of table T^l restricted to rows t^l_i such that $v(t^l_i, a^\star) = v^\star$ (that is: the conditional probability distribution of the latent space given that $v(t^l_i, a^\star) = v^\star$, the set of such observations and the parameters $\boldsymbol{\theta}$). Likewise, $\phi^l_{\text{enc}}(t^l_i)$ is the conditional probability of the latent representation of row t^l_i given the parameters and observations. Decoder $\phi^l_{\text{dec}}(t^l_i)$ is defined analogously. $P_{(l, v^\star)}$ is the prior for table T^l and unique attribute $v^\star \in V^\star$ represented in the latent space (in our case each chosen as multivariate normal Gaussian); β_l is a table-specific factor analogous to standard β in β-VAE.

Then, we extract the rounded values $n^l_1, \ldots, n^l_n$ of the attribute a_{T^l} from $\hat{T}'^\star$, with $n = |\hat{T}'^\star|$. Each n^l_k represents the number of rows in the synthetic

secondary table $\hat{T}_l$ that should refer to the k-th primary entity in $\hat{T}^\star$. We also take the corresponding points $\hat{\boldsymbol{y}}^\star_1, \dots, \hat{\boldsymbol{y}}^\star_n$ in the primary latent space.

Subsequently, for each $k \in \{1, \dots, n\}$, we:

(1) compute $Y'^{l,k}$ as the image $\phi^l_{\text{enc}}(X^{l,k})$ in the reduced latent space of $X^{l,k}$, where $X^{l,k}$ is the sub-collection of points in X^l that corresponds to the k-th primary entity in the primary table, i.e., $X^{l,k} = \{\boldsymbol{x}^l_i : v(t^l_i, a^\star) = k\}$;

(2) fit a multivariate normal distribution over $Y'^{l,k}$ obtaining $\boldsymbol{\mu}^{l,k}$ and $\boldsymbol{\Sigma}^{l,k}$;

(3) sample the multivariate normal distribution $\mathcal{N}\left(\boldsymbol{\mu}^{l,k}, \boldsymbol{\Sigma}^{l,k}\right)$ n^l_k times obtaining $\hat{Y}'^{l,k}$;

(4) build $\hat{Y}^{l,k}$ by row-wise concatenation of $\hat{Y}'^{l,k}$ and n^l_k copies of $\hat{\boldsymbol{y}}^\star_k$, that constitute the disentangled variables representing the k-th primary entity;

(5) compute $\hat{X}^{l,k}$ as the image of $\phi^l_{\text{enc}}(\hat{Y}^{l,k})$.

Once, we collected all the h collections $\hat{X}^{l,1}, \dots, \hat{X}^{l,h}$, we obtain $\hat{X}^l$ by merging them, i.e., $\hat{X}^l = \bigcup_{k=1}^{k=n} \hat{X}^{l,k}$. Finally, we obtain $\hat{T}'^l$ by applying ϕ^{-1}_{type} on $\hat{X}^l$ and then post-process $\hat{T}'^l$ by adding the attribute $a^\star$: for each row, we set the value to the k corresponding to the $X^{l,k}$ the row comes from.

Summarizing, for each secondary table T^l we do the following steps in order to produce the corresponding synthetic secondary table $\hat{T}^l$:

$$T'^l := \text{preProc}\left(T^l\right) \tag{4.44}$$

$$X^l := \phi^l_{\text{type}}\left(T'^l\right) \tag{4.45}$$

$$\left(\phi^l_{\text{enc}}, \phi^l_{\text{dec}}\right) := \text{learn}\left(X^l\right) \tag{4.46}$$

$$X^{l,k} := \left\{\boldsymbol{x}^l_i : v\left(t^l_i, a^\star\right) = k\right\} \qquad \forall k \tag{4.47}$$

$$Y'^{l,k} := \phi^l_{\text{enc}}\left(X^{l,k}\right) \qquad \forall k \tag{4.48}$$

$$\left(\boldsymbol{\mu}^{l,k}, \boldsymbol{\Sigma}^{l,k}\right) := \mathcal{N}^{-1}\left(Y^{l,k}\right) \qquad \forall k \tag{4.49}$$

$$\hat{Y}'^{l,k} :\overset{n^l_k}{\sim} \mathcal{N}\left(\boldsymbol{\mu}^{l,k}, \boldsymbol{\Sigma}^{l,k}\right) \qquad \forall k \tag{4.50}$$

$$\hat{Y}'^{l,k} := \left[\hat{Y}'^{l,k} \text{ nOf } \left(\hat{\boldsymbol{y}}^\star_k, n^l_k\right)\right] \qquad \forall k \tag{4.51}$$

$$\hat{X}^{l,k} := \phi^l_{\text{dec}}\left(\hat{Y}^{l,k}\right) \qquad \forall k \tag{4.52}$$

$$\hat{X}^l := \bigcup_{k=1}^{k=n} \hat{X}^{l,k} \tag{4.53}$$

$$\hat{T}'^l := \phi^{-1}_{\text{type}}\left(\hat{X}^l\right) \tag{4.54}$$

$$\hat{T}^l := \text{postProc}\left(\hat{T}'^l\right) \tag{4.55}$$

We remark that, for secondary tables, the number of rows in the table is not chosen by the user, but is instead generated as part of the synthetic data. Hence,

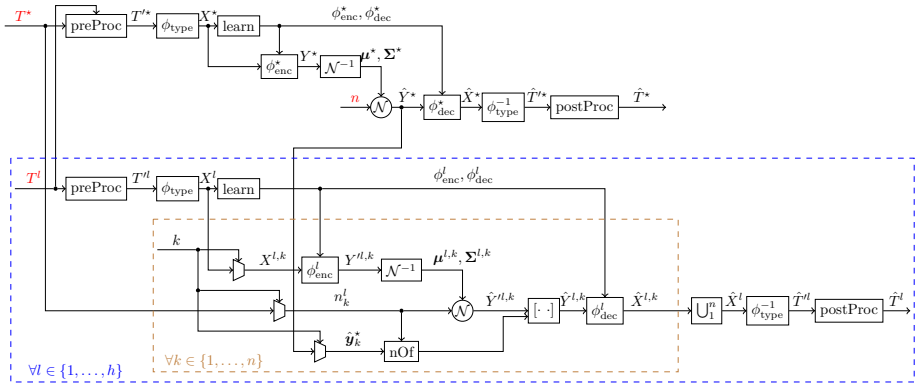


Figure 4.2: Schematic representation of the steps for going from a relational dataset $D = (T^\star, T^1, \ldots, T^h)$ to a compatible synthetic relational dataset $\hat{D} = (\hat{T}^\star, \hat{T}^1, \ldots, \hat{T}^h)$. In red, the parameter and inputs provided by the user; in colored dashed rectangles, parts of the steps that are repeated for l and k.

we meet the first goal stated in Section 4.4.3. Moreover, since we reserve a portion of the latent space for disentangled variables that represents the primary entity, and since we use several copies of the same values, we meet the second goal stated in Section 4.4.3: rows in a secondary table that refers to the same primary entity are generated from point in the latent space that are the same for a portion of the dimensions.

The overall process for generating synthetic relational datasets is depicted in Figure 4.2.

4.4.4 Implementation

In practice, we implemented the approach described above based on a mix of existing (e.g., TensorFlow) and ad hoc developed software frameworks. We made the synthetic data generation service available online, as a Software-as-a-Service (SaaS) tool. If particular security concerns forbid the transfer of data to the cloud, the structure could also be deployed directly on a client local infrastructure.

We defined an API for accessing the service. When the user requires a synthetic data generation service, he/she uploads the data through the API. We store the data using an array of HDFS nodes. A centralized component of the system, that we call coordinator, saves the data structure, the targeted section of the HDFS, the TensorFlow model (i.e., the various parameters $\boldsymbol{\theta}$ for the VAEs), and the job details on a job database. Computing nodes, equipped with GPUs and having direct access to HDFS, train the TensorFlow model. After training, compute nodes generate the requested amount of synthetic data and store them to HDFS. The coordinator also takes care of training and generating

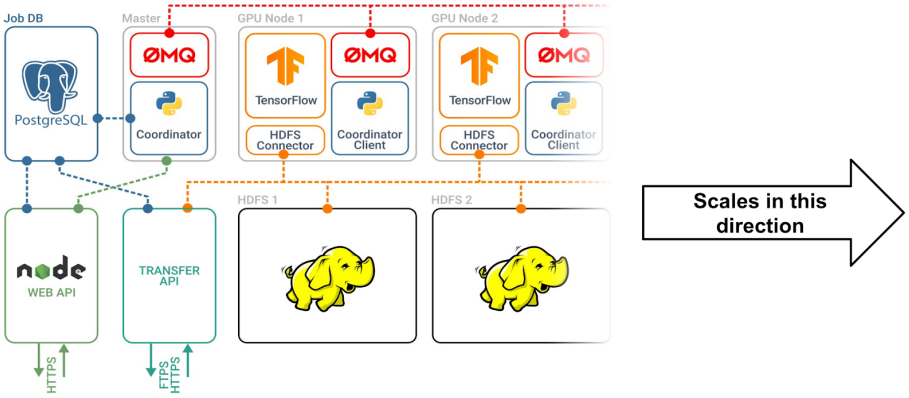


Figure 4.3: Overview of the implementation as a SaaS.

jobs and of the resource allocation for each of them, using a message queue. The architecture is depicted in Figure 4.3.

4.5 Conclusion

In this chapter, we identified the variational autoencoder (VAE) framework as a viable method for generating synthetic tabular and relational data that is both utility preserving and disclosure averse. That is: it retains the analytic utility of a real dataset, but also protects the privacy of the real dataset's data subjects. To quantify the degree to which these two objectives are met, we introduced normalized performance metrics for synthetic data. We then introduced data preprocessing techniques that map raw data into data suitable for the application of VAE-based methods. These include techniques for processing real-valued numeric, discrete numeric, time, categorical and binary attributes realistically.

A particular novelty of our method is that it can also accurately reproduce the distributions of missing values. We introduce three types of VAE for tabular data (VAE, β-VAE and introspective VAE) and one for treating relational data (multi-layer VAE). We also describe how to apply the performance metrics in the presence of advanced data types. An overview of an architecture for the implementation of the methods was also provided. Combined, these results indicate that VAE can be applied for generation of synthetic tabular and relational data, and metrics are available to evaluate the results.

Chapter 5

Experimental Evaluation of Generative Models

In this chapter, we answer the research question *"How well do VAE-based methods for generating synthetic tabular and relational data perform in practice?"*. To answer this question, we apply the developed methods for tabular data to publicly available datasets. The datasets vary in size, combinations of data types and application (the ML problem to be applied to the synthetic data after it is generated). This casts light on the versatility of the methods. We test the method for relational data with a custom randomly generated relational dataset. The experiments are entirely reproducible. Experimental results are presented in a clear format and discussed in detail.

Dataset	$\|T\|$	$\|A(T)\|$	R+D	B	C	T	Target
Insurance	1338	14	14	0	0	0	R
Covertype	581 012	54	10	44	0	0	C
Bank	45 211	16	3	3	8	2	C
Relational							
Users	1000	4	4	0	0	0	C
Events A	20 050	3	3	0	0	0	
Events B	20 253	2	2	0	0	0	

Table 5.1: Overview of the four datasets used in the experiments, one row per table. Column $|T|$ indicates the number of rows in the table. Column $|A(T)|$ indicates the number of attributes. Columns R+D, B, C, and T indicate the number of real-valued and discrete, binary, categorical, and time attributes respectively. The column Target indicates whether the ML problem build on the table is a classification (C) or a regression (R) problem.

5.1 Datasets

We took three single-table datasets from the UCI Machine Learning Repository [77]: Insurance, Covertype, Bank, The size of the corresponding tables ranges from ≈ 1000 to $\approx 600\,000$ rows. The number of attributes ranges from 14 to 54: overall, all types of attributes are represented (real-valued, discrete, time, categorical, binary). Each dataset has a predefined target attribute, i.e., one for which a supervised learning problem can be built: for two of them (Covertype, Bank), the target attribute is categorical; for Insurance, it is numeric. The corresponding problems are hence classification and regression.

We also built an ad hoc relational dataset composed of three tables (Users, Events A, Events B): Users is the primary table. We here included only numerical attributes, with the exception of a single categorical attribute, for the table Users, that we set as the target attribute.

Table 5.1 summarizes the salient information of the four datasets.

5.2 Procedure

We performed a five-fold cross validation of our data synthesis technique. That is, for each dataset, we split the dataset in 5 partitions. Then, for each partition D_{out}, we took it apart and learned a synthetic dataset $\hat{D}$ on the four remaining partition D. Finally, we measured the UP and DA indexes on the pair $\hat{D}, D$. For the single-table datasets, we perform the split in partitions by simply partitioning the corresponding tables in equally sized slices (after shuffling the rows) For the relational dataset, we first partitioned the primary table in equally sized slices, then, for each partition, we selected the rows of the two secondary tables accordingly—as a results, the overall size of the five datasets were different.

We applied our data synthesis pipeline in the three variants (VAE, β-VAE, IntroVAE) for each partition of the single-table datasets. We used the single proposed variant for the relational dataset partitions. Overall, we hence generated $3 \cdot 3 \cdot 5 + 1 \cdot 1 \cdot 5 = 50$ synthetic datasets.

Concerning the UP and DA indexes, we proceeded as follows.

For MC and classification datasets, we considered Random Forest (RF), logistic regression classifier (LRC), adaptive boosting (ADA), and multi-layer perceptrons (MLP) as solutions (i.e., learning techniques) and accuracy (Acc.) and area under the ROC curve (AUC) as effectiveness metrics. For MC and regression datasets (only Insurance), we considered linear regressions (LR), ridge regression (RR), polynomial Support Vector Regressions (SVR), and multi-layer perceptrons (MLP) as solutions and the coefficient of determination (R^2) and the mean squared error (MSE) as effectiveness metrics. For all single-table datasets, we used all the attributes as features when applying the learning technique for building a prediction model for the target attribute. For the relational dataset, we (i) built a single aggregated table by associating with each row of the User table a tuple consisting of the means of the attributes of the corresponding rows in the secondary table, hence obtaining a table with 10 attributes, and (ii) use all the attributes of the aggregated table as features. For clarity, we present these indexes as triplets: e.g., MC-RF-AUC is the MC measured on a classification datasets using RF as learning technique and AUC for evaluating the learned models.

For PCD, we computed the Pearson correlation on the numerical conversion of the tables, rather than on the actual tables, i.e., on $X = \phi_{\text{type}}(T)$ and $\hat{X} = \phi_{\text{type}}(\hat{T})$ rather than on T and $\hat{T}$. This way, we circumvented the problem of finding a suitable correlation measure for each possible pair of attribute types.

For CSE, we proceeded as for PCD. Moreover, for the relational dataset case, we measured this index on the (numerical conversion of the) single aggregated table. We use k-means as clustering technique, choosing the value of k automatically in each case with the elbow method.

Finally, for all the NNDR indexes (NNDR_μ, NNDR_σ, and NNDR_p) and for NNDD, we proceeded as for CSE. For NNDD, we performed the statistical significance test with two values of α, 0.05 and 0.01.

We performed all the experiments by using the implementation of our pipeline described in Section 4.4.4 for generating the datasets and an ad hoc piece of software (written in Python 3) for measuring the UP and DA indexes on the generated datasets. We set the parameters of the synthesis pipeline to the values shown in Table 5.2 after some preliminary experiments.

5.3 Results and Discussion

Tables 5.3 and 5.4 present the results, respectively for the classification datasets, for which the target variable is categorical (Covertype, Bank, and the relational one), and for the regression dataset, for which the target variable is numeric (Insurance)—we show the results in two separate tables because the ML tech-

Param	Context	Description	Value
k	ϕ_{type} R+D	N. of quantiles	$\min(1000, \lvert T \rvert)$
δt	ϕ_{type} T	Time span of the interval	$\frac{1}{1000}(\tau_{\max} - \tau_{\min})$
r	VAE (all)	Latent space dimension	10
β	β-VAE	Reconstruction accuracy vs. prior similarity trade-off	0.6
β_l	ML-VAE	Reconstruction accuracy vs. prior similarity trade-off	0.3

Table 5.2: Summary of the parameter values.

		Utility-preservation (UP)										Disclosure-averseness (DA)				
		MC-RF		MC-LRC		MC-ADA		MC-MLP				NNDR			NNDD	
Dataset	Method	Acc.	AUC	Acc.	AUC	Acc.	AUC	Acc.	AUC	PCD	CSE	μ	σ	p	0.05	0.01
Covertype	VAE	0.12	0.05	0.14	0.06	0.11	0.10	0.26	0.19	0.00	0.03	0.02	0.95	0.16	0	0
	β-VAE	0.17	0.03	0.05	0.06	0.10	0.07	0.03	0.20	0.01	0.08	0.19	0.83	0.02	0	0
	IntroVAE	0.17	0.04	0.10	0.05	0.10	0.09	0.08	0.20	0.02	0.17	0.32	1.14	0.01	0	0
Bank	VAE	0.10	0.12	0.29	0.12	0.10	0.22	0.08	0.12	0.02	0.04	0.22	0.75	0.25	0	0
	β-VAE	0.01	0.05	0.01	0.03	0.03	0.07	0.03	0.06	0.00	0.04	0.06	0.87	0.00	4	4
	IntroVAE	0.00	0.05	0.02	0.04	0.03	0.08	0.02	0.06	0.01	0.09	0.25	0.92	0.00	3	4
Relational	ML-VAE	0.04	0.16	0.08	0.23	0.17	0.03	0.03	0.23	0.08	0.18	0.10	0.43	0.02	0	0

Table 5.3: Results for the classification datasets. For each of the rightmost 13 indexes (from MC-RF-Acc. to NNDR_p), the table shows the mean value across the five folds. For the two remaining indexes (NNDD with $\alpha = 0.05$ and $\alpha = 0.01$), the table shows the number of folds for which the statistical test rejects the null hypothesis. For all the indexes, the lower the better.

niques and the effectiveness metrics used while evaluating MC are different for the classification (Accuracy and AUC) and regression cases (R^2 and MSE). Both tables show the variants of our method applied to different datasets on rows and the values for the performance indexes on columns, grouped in UP indexes (columns 3 to 13) and DA indexes (last 5 columns). For each row, i.e., for each pair of synthetic data generation technique and dataset, the tables show the mean value across the five folds of each performance index, with the exception of the last two indexes, NNDD computed for $\alpha = 0.05$ and for $\alpha = 0.01$: for those, the tables show the number of folds for which the statistical test rejects the null hypothesis that the two distributions are the same (see Section 4.3.3). For all the indexes, the lower the better.

By looking at the table, it can be seen that, in general, all UP values are very low, i.e., very good. In particular, for what concerns MC, β-VAE and IntroVAE obtain better results than the vanilla VAE in almost all cases (i.e., dataset and effectiveness metric), one exception being MC-RF-Acc. for Covertype. For the

		Utility-preservation (UP)										Disclosure-averseness (DA)				
		MC-LR		MC-RR		MC-SVR		MC-MLP				NNDR			NNDD	
Dataset	Method	R^2	MSE	R^2	MSE	R^2	MSE	R^2	MSE	PCD	CSE	μ	σ	p	0.05	0.01
Insurance	VAE	0.06	0.54	0.06	0.52	0.15	1.09	0.14	1.14	0.01	0.08	1.11	4.17	0.02	0	2
	β-VAE	0.04	0.29	0.04	0.29	0.10	0.39	0.11	0.42	0.01	0.14	0.61	2.82	0.05	0	3
	IntroVAE	0.03	0.21	0.03	0.21	0.20	0.48	0.09	0.39	0.02	0.09	1.10	4.29	0.04	2	3

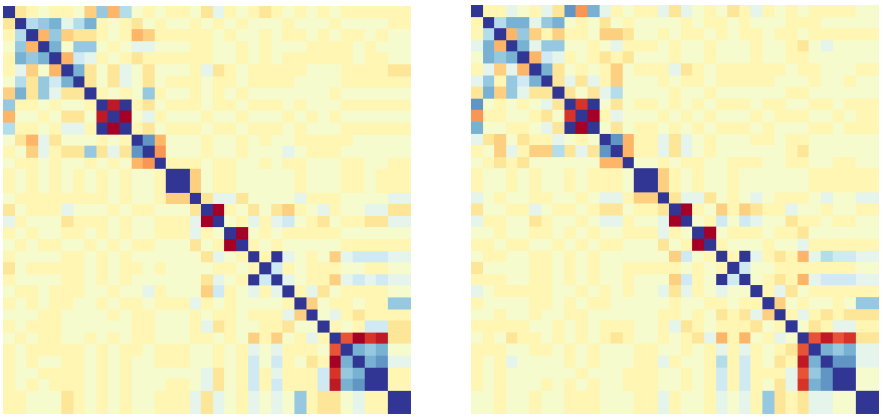
Table 5.4: Results for the regression dataset. For each of the rightmost 13 indexes (from MC-LR-R^2 to NNDR$_p$), the table shows the mean value across the five folds. For the two remaining indexes (NNDD with $\alpha = 0.05$ and $\alpha = 0.01$), the table shows the number of folds for which the statistical test rejects the null hypothesis. For all the indexes, the lower the better.

regression dataset, Insurance, the absolute values for MC-*-MSE are larger than the other MC values: we believe that this is because MSE does, differently from the other effectiveness measures, depend on the scale of the target variable. However, the relative ranking among the three variants of VAEs stays the same.

Concerning PCD, it can be seen that the values are very close to zero in all the cases. This means that the way the variables are correlated in the original and synthetic dataset is very similar. For gaining further insights in how our technique correctly preserve the general structure of the data, in terms of correlation, we show in Figure 5.1 the correlation matrices for the original and synthetic data (after the numerical conversion, i.e., on X and $\hat{X}$) for one fold of the Bank dataset with β-VAE. The figure highlights that the pairwise inter-dependencies among variables of the original dataset are well preserved in the synthetic dataset.

In terms of differences among the three variants, the values of CSE, differently from those of PCD, seem to suggest that IntroVAE is in general worse than the other two variants: this means that the synthetic data generated with this technique cluster in a way that is slightly different than the way in which the original data do. Overall, the values for UP performance indexes suggest that all the four technique variants (VAE, β-VAE, IntroVAE, and ML-VAE for the relational dataset case) are able to generate synthetic data that can be useful for replacing the original data.

Concerning the DA indexes, the numbers in Tables 5.3 and 5.4 suggest that, for classification datasets IntroVAE, seems in general better in generating data that prevents disclosure: NNDR$_p$ for this variant is consistently lower than the values of the index for the other two variants. We recall that the closer NNDR$_p$ to 0, the more even the distribution between original rows that are more similar to other original rows and to synthetic rows. In other words, with NNDR$_p = 0$, for any original row, the probability that its closest row is a synthetic data point is exactly 50 %: hence, it is hard to tell apart original and synthetic rows by just looking at proximity to known rows. The other two indexes related to the concept of nearest neighbor distance ratio (NNDR$_\mu$ and NNDR$_\sigma$) appear more difficult to interpret. For the regression dataset, the values of NNDR$_p$ are in



(a) Original data.

(b) Synthetic data.

Figure 5.1: Correlation matrices (of variables after the numerical conversion) of original and synthetic data for one fold of the Bank dataset with β-VAE. Red denotes negative correlation; blue denotes positive correlation; yellow denotes ≈ 0 correlation.

general very close to 0, regardless of the VAE variant, but the values for NNDR_μ are larger. Since the fact that the target variable does not impact in any way on neither the generation process, nor on the measurement of DA indexes, we infer from this observation that Insurance is a harder dataset, from the point of view of DA. For what regards NNDD, that builds over the same idea of distance, it can be seen that the results are very good (NNDD $= 0$ for both values of α) for all the variants and datasets, with the exception of Bank, for which β-VAE and IntroVAE do not obtain the best score. An example of the cumulative distributions of minimal distances between pairs of real data points and pairs of one real and one fake data point is provided in Figure 5.2.

Finally, Table 5.3 shows that the ML-VAE works well with the relational dataset, since all DA indexes have low values.

We performed our experiments on a machine with an Intel i7 6500U CPU with 4 logical cores at 2.5 GHz, equipped with 16 GB RAM and a NVidia RTX1080 GPU. The generation of the synthetic dataset took a time that was dependent mostly on the size of the original dataset: on average, 0.02 s for each row.

5.4 Conclusion

In this chapter, we evaluated experimentally our VAE-based methods for synthetic generation of realistic tabular and relational data. We used performance metrics to evaluate the methods' performances on three publicly available

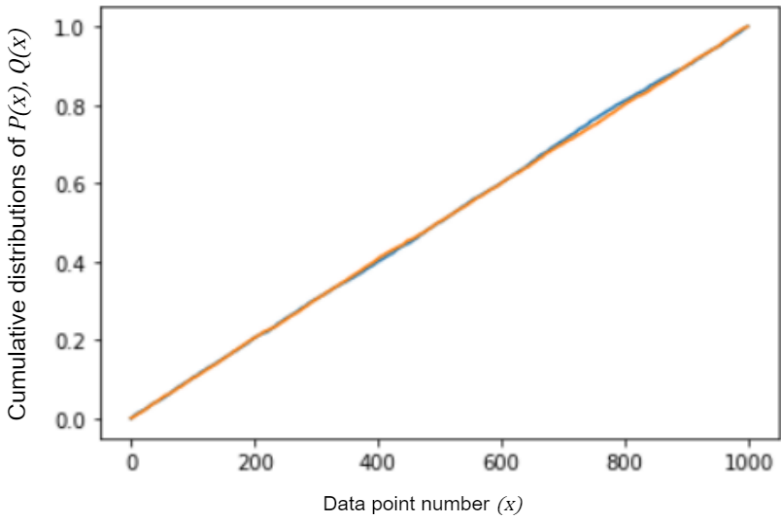


Figure 5.2: Cumulative distributions of minimal distances between pairs of two real points (orange) and between pairs of one real and one fake point (blue). The supremum of the distance between these distributions is the test statistic for NNDD.

datasets consisting of one table each and on a custom relational dataset consisting of three tables. The results indicated that β-VAEs and IntroVAEs are particularly reliable for generating synthetic tabular data. They preserve privacy and vital statistical data properties such as correlations. Furthermore, they preserve utility for the development of ML models, particularly in the context of classification problems. For regression problems, the utility appears more difficult to guarantee. For relational data, ML-VAE showed effective in generating reliable and privacy preserving synthetic data.

Chapter 6

Conclusion and Future Research

6.1 Conclusion

In this thesis, we studied synthetic data generated through deep-learning methods as a privacy enhancement technology. Our core research objective was to develop algorithms that produce *utility preserving* and *disclosure averse* synthetic data modeled after a real input dataset. Utility preservation means that when the synthetic data is used to train artificial intelligence models, the results are on a par with the results of such methods applied to the real dataset. Disclosure averseness means that access to the synthetic dataset does not provide the synthetic data owner with sensitive information about any of the data subjects in the real dataset.

To achieve this objective, we first provided an overview of machine learning. This provided an overview of terminology that was later applied. However, through practical case studies, it also illustrated that there is a pronounced demand for AI methods in industry. Such methods can only be applied if full and secure data access is available.

Next, we studied the legislative framework for data privacy, including existing privacy technologies and their legal status. This study revealed that the number of data breaches and their costs are increasing rapidly in the European Union. The most common tyoes of breaches are *re-identification attacks* and *linkage attacks*. In the former, an attacker can identify a data subject, even if the victim's record is anonymized in a dataset. In the latter, the attacker can link victim's records from two distinct databases, in one of which the attacker knows the victim's identity. By doing so, the attacker can infer sensitive information. *Parameter inference attacks*, in which the attacker aims to infer the value of a specific attribute were rare in practice, because they are difficult to formally define.

Traditional data anonymization techniques rely on data distortion. Besides

not providing sufficient protection, they reduce the degree to which the data accurately represent reality. Differential privacy is more reliable, both from a utility and a privacy viewpoint. However, by providing merely aggregate data, they severly limit the number of use cases that can be conducted, excluding AI methods. Synthetic data is an emerging technology for which a thorough legal evaluation is still lacking. The technology is, however, met with enthusiasm from AI practitioners, domain experts and policy-makers. By removing any link between data and real individuals, synthetic data do not disclose any sensitive information. Through deep learning methods, such data can, however, be constructed to contain the same analytic utility as any real dataset.

We then studied deep-learning generative models as a privacy enhancing technology for tabular and relational data. By introducing novel data preprocessing methods, we demonstrated that variational autoencoder models originally conceived for image generation can be applied in this context. We also introduced performance metrics to measure the degrees of utility preservation and disclosure averseness of synthetic datasets modeled after specific real datasets. We also provided remarks on efficient implementation of the methods, in a manner that can handle large quantities of data.

Finally, we conducted experiments to assess how well the designed methods perform. We used three publicly available datasets with highly distinctive numbers of records, combinations of attribute types and applications (classification and regression) to evaluate the methods for tabular data. For relational data, we used a randomly generated dataset with one parent table and two child tables. The results indicated that the methods can accurately produce synthetic data for classification and regression tasks.

6.2 Future Research

As synthetic data is an emerging field, it currently lacks jurisprudence. Future research should address the technology's legal status, imposing conditions on its use. This type of research is multi-disciplinary in nature, as it requires knowledge of law, artificial intelligence, application domains (such as healthcare and finance), and possibly ethics and philosophy. Furthermore, while re-identification and linkage attacks have sound definitions, the definition of a parameter inference attack is more opaque. Additional multi-disciplinary research should address this, possibly redefining the term and introducing quantitative metrics to assess the occurrence and gradation of attacks. Future research should inspect whether the target application of synthetic data can inform the performance metrics used, and thereby the hyperparameter optimization process. This can help tune parameters so that VAE-based models are better equipped to generate synthetic data for regression problems. Metrics specifically developed for measuring the performance of generators of relational data should also be further inspected.

In this these, we noted that some metrics cast light on both privacy protection and utility preservation. Intuitively put, this is because if a synthetic

dataset is very realistically distributed over its space, its data points will not be centered suspiciously around specific empirical (real) data points. As such, the synthetic data points will not leak information about any empirical data points. The accurate distribution will, however, also increase the utility of the synthetic data for statistical and artificial intelligence purposes. This observation should be studied further, as utility and privacy are often seen as competing objectives.

About the Author

Academic background I obtained both a bachelor's and a master's degree in mathematical optimization from the Sapienza University in Rome. I also obtained a master's degree in Operations Research at Maastricht University (The Netherlands). This field lies in the interplay between mathematics, AI, computer science and economics. While I still rely on the mathematical maturity I gained during my studies, this was only part of what I learned. I also learned how to function as part of an interdisciplinary team (through the university's extracurricular program MARBLE for talented students); how to set up a strong network and how to properly launch and manage a company (through the university's extracurricular program LaunchBase). I learned that in life and business, there are no mistakes, there are only opportunities!

Professional experience I started my professional career in The Netherlands, where I worked for over two years as a data scientist at the Medtronic Bakken Research Centre. Subsequently, I joined Allianz Technology in Trieste, where I was the principal data scientist in the business development team. In both these positions, I witnessed first-hand how big the impact of data science is. By properly analysing data, not only can business targets be much easier achieved, lives can even be saved. It was through these experiences that I shaped my vision to make cutting-edge advances in AI and data science available to the world.

Scientific career and founding of Aindo To realise this vision, I knew I needed two things: advanced expertise in AI and a vessel for my natural entrepreneurial and networking talents. For this reason, I commenced my studies as a PhD researcher in AI at the University of Trieste in 2017. In 2018, I followed my entrepreneurial instincts and founded Aindo. As a PhD scholar, my research interests soon became centered around deep generative models. As an entrepreneur, I then quickly realised that there was a tremendous potential for

such models as an enabling technology for AI innovation. My research therefore quickly became intertwined with the development of a synthetic data product at Aindo.

What I love about my work is that it allows me to combine my passion for data science with a friendly and engaging approach to business. This combination has proven to be rather successful: while still an active scientific researcher at the University of Trieste, I was able to grow Aindo into a company with departments spanning R&D, engineering, IT, business intelligence and communications. I myself am mainly involved in R&D, engineering and business development. This includes designing and implementing new products and solutions, but also positioning them in a complex market and fostering a strong network.

Achievements The Aindo synthetic data platform was conceived in late 2019 and soon gained traction with industry experts, investors and organizations. In January 2020, it was one of eight out of roughly five hundred companies to win the European Data Incubator (EDI). Aindo was granted €100 000 for the further development of the product. After testing a proof of concept with multiple industrial partners in April 2020, the EDI trajectory was successfully completed in September 2020, with the implementation of a fully functional beta-version. In June 2020, Aindo was one of ten finalists at the COLLISSION 2020 conference, one of North America's most influential tech conferences. In October 2020, the software was tested on the Reachout platform. In November 2020, Aindo was selected as a participant in Oxford University's prestigious Creative Destruction Lab (CDL) for the AI stream of the academic year 2020-2021. *Wired Italia*, *Le Scienze*, *The MIT Technology Review*, and *Il Sole 24Ore* have all dedicated articles to Aindo's succes story.

In late 2021, Aindo obtained a €2.8M investment from venture capitalists. This investment allows us to step up all of our activities. At fifteen people, we are currently assembling one of the best AI teams in Europe, if not the world. We are upscaling our R&D activities so that our technology is always the most cutting-edge solution. We are increasing our IT activities to facilitate the most intuitive user-experience. Our business development and communications branches are quickly expanding, so that we can better inform the world about the endless possibilities that deep learning generative models offer.

Appendix A

Assessment of Performance Metrics

A.1 Experiments: Cluster Symmetric Evenness

In this section, we experimentally determine the extend to which the CSE metric effectively measures utility preservation (UP) between datasets. To do so, we procede as follows:

1. We construct pairs (D_1, D_2) of datasets in $I_1 \times I_2$, for $I_1 = [1800, 2000]$ and $I_2 = [1000, 1200]$ at random;

2. Each dataset has strong clustered behavior. That is: there are k clusters present in each dataset. this is achieved by randomly smapling k "means" $\mu_1, \mu_2, ..., \mu_k$ from the uniform distributions over the intervals I_1 and I_2;

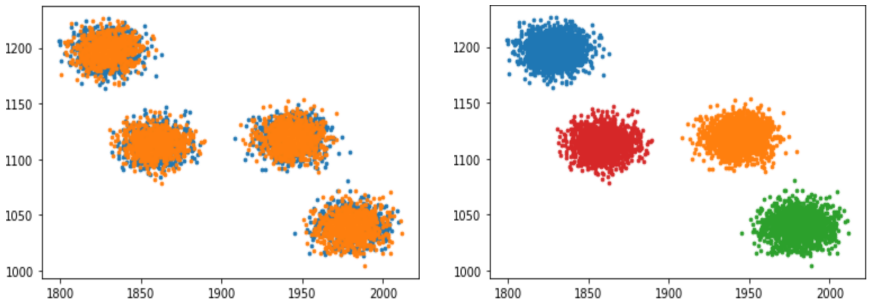
3. Next, we treat the obtained means as the means of normal distributions $\mathcal{N}(\mu_k, 2), k = 1, 2, 3, ...$ and we sample a number in $\{700, 701, ..., 1000\}$ of points (selected by sampling from $U[700, 1000]$) from each normal distribution;

4. We compute the CSE for D_1 and D_2;

5. We repeat this process in three types of experiments:

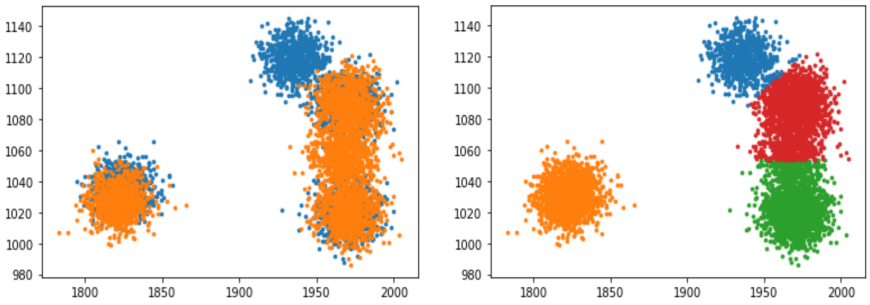
   - Equal sampling: the normal distributions of D_1 and D_2 have the same means;
   - Half-equal sampling: $\lceil k2 \rceil$ distributions of the datasets D_1 and D_2 have the same means; the other $\lfloor k2 \rfloor$ have randomly selected means that are distinct between sets D_1 and D_2;
   - Unequal sampling: All $2k$ means distinct.

If the cluster symmetric evenness is an accurate metric of UP, then it will be close to zero under equal sampling; close to 0.5 under half-equal sampling; and close to one under unequal sampling. Figures A.1, A.2 and A.3 show the results of one sampling trial for equal, half-equal and unequal sampling types, respectively. Table A.1 contains experimental results for 100 experiments per sampling type. The results indicate that the CSE is indeed close to 0 for equal sampling; close to 0.5 for half-equal sampling; and close to 1 for unequal sampling. The larger standard deviation for the latter two sampling types is because when distributions are randomly generated "unequally", there might still be considerable overlap, leading small CSE results.



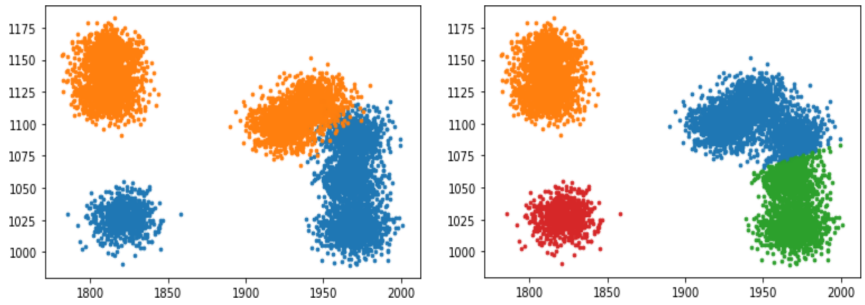
(a) D_1 (orange) versus D_2 (blue) (b) Clustered data

Figure A.1: Equal sampling results: CSE= 0.07



(a) D_1 (orange) versus D_2 (blue) (b) Clustered data

Figure A.2: Half-equal sampling results: CSE= 0.52



(a) D_1 (orange) versus D_2 (blue) (b) Clustered data

Figure A.3: Unequal sampling results: CSE= 0.90

Data pair	Equal		Half-equal		Unequal	
Statistic	μ	σ	μ	σ	μ	σ
Value	0.06	0.02	0.61	0.17	0.87	0.10

Table A.1: Experimental results for 100 experiments per data pair type. Each dataset has four means. The number of datapoints per dataset is randomly sampled from $U\,[700, 1000]$.

Bibliography

[1] The European Comission. Shaping Europe's Digital Future, 2019. URL https://ec.europa.eu/info/strategy/priorities-2019-2024/europe-fit-digital-age/shaping-europe-digital-future_en.

[2] The European Comission. Towards a common European data space, 2018. URL https://eur-lex.europa.eu/legal-content/en/TXT/?uri=CELEX:52018DC0232.

[3] Luc Rocher, Julien M Hendrickx, and Yves-Alexandre De Montjoye. Estimating the success of re-identifications in incomplete datasets using generative models. *Nature communications*, 10(1):1–9, 2019.

[4] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, 2016.

[5] Huaibo Huang, Ran He, Zhenan Sun, Tieniu Tan, et al. Introvae: Introspective variational autoencoders for photographic image synthesis. *Advances in Neural Information Processing Systems*, 31, 2018.

[6] Noseong Park, Mahmoud Mohammadi, Kshitij Gorde, Sushil Jajodia, Hongkyu Park, and Youngmin Kim. Data synthesis based on generative adversarial networks. *Proceedings of the VLDB Endowment*, 11(10):1071–1083, 2018.

[7] Daniele Panfilo, Cosimo De Nunzio, Antonio Luigi Pastore, Sebastiano Saccani, Alexander Boudewijnb, Pietro Tortella, Manuela Mattioli, Riccardo Lombardo, Antono Carbone, Andrea Fuschi, Lorenzo Dutto, Joern Witt, Eric Medvet, and andrea Tubaro. Using Machine Learning Tools to Predict Prostate Cancer Upgrading After Robotic Radical Prostatectomy. *The Journal of Urology*, 201(4S):1098, 2019.

[8] Gianfranco Fenu, Eric Medvet, Daniele Panfilo, and Felice Andrea Pellegrino. Mosaic image segmentation using u-nets. In *9th International Conference on Pattern Recognition Applications and Methods (ICPRAM), held in Valletta, Malta*, pages 485–492. Scitepress, 2020.

[9] T. Mitchell. *Machine Learning*. McGraw Hill, 1997.

[10] Trevor Hastie, Robert Tibshirami, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference and Prediction, second edition*. Springer Series in Statistics, 2017.

[11] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. The MIT Press, Cambridge, Massachusettts, 2016.

[12] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction, second edition*. The MIT Press, Cambridge, Massachusettts, 2018.

[13] I.T. Jolliffe. *Principal Component Analysis*. Springer Series in Statistics, 2002.

[14] Sheldon M. Ross. *Introduction to Probability Models*. Academic Press: Waltham, MA, 1972.

[15] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. Introduction to information retrieval. Cambridge University Press, 2008.

[16] Tin Kam Ho. Random decision forests. *Proceedings of 3rd International Conference on Document Analysis and Recognition*, 2002.

[17] Raúl Rojas. Adaboost and the super bowl of classifiers a tutorial introduction to adaptive boosting, 2009.

[18] Yann LeCun. *Generalization and Network Design Strategies. Technical Report.* University of Toronto, 1989.

[19] D. Rumelhart, G. Hinton, and R. Williams. Learning representations by backpropagation errors. *Nature*, 323:533 – 536, 1986.

[20] Yannis Assael, Thea Sommerschield, and Jonathan Prag. Restoring ancient text using deep learning: a case study on greek epigraphy. *arXiv preprint arXiv:1910.06262*, 2019.

[21] Moritz Neumüller, Andreas Reichinger, Florian Rist, and Christian Kern. 3d printing for cultural heritage: Preservation, accessibility, research and education. In *3D Research Challenges in Cultural Heritage*, pages 119–134. Springer, 2014.

[22] Lamia Ben Youssef and Stéphane Derrode. Tessella-oriented segmentation and guidelines estimation of ancient mosaic images. *Journal of Electronic Imaging*, 17(4):043014, 2008.

[23] Alberto Bartoli, Gianfranco Fenu, Eric Medvet, Felice Andrea Pellegrino, and Nicola Timeus. Segmentation of mosaic images based on deformable models using genetic algorithms. In *International Conference on Smart Objects and Technologies for Social Good*, pages 233–242. Springer, Springer, 2016.

[24] Radu Comes, Zsolt Buna, and Ionut Badiu. Creation and preservation of digital cultural heritage. *Journal of Ancient History and Archaeology*, 1(2), 2014.

[25] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[26] Luc Vincent and Pierre Soille. Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(6):583–598, 1991.

[27] L Fazio, M Lo Brutto, and G Dardanelli. Survey and virtual reconstruction of ancient roman floors in an archaeological context. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 42(2/W11), 2019.

[28] Mariapaola Monti and Giuseppe Maino. Image processing and a virtual restoration hypothesis for mosaics and their cartoons. In *International Conference on Image Analysis and Processing*, pages 486–495. Springer, 2011.

[29] Lamia Benyoussef and Stéphane Derrode. Analysis of ancient mosaic images for dedicated applications. In Filippo Stanco, Sebastiano Battiato, and Giovanni Gallo, editors, *Digital Imaging for Cultural Heritage Preservation: Analysis, Restoration, and Reconstruction of Ancient Artworks*. CRC Press, 2011.

[30] Thorsten Falk, Dominic Mai, Robert Bensch, Özgün Çiçek, Ahmed Abdulkadir, Yassine Marrakchi, Anton Böhm, Jan Deubner, Zoe Jäckel, Katharina Seiwald, et al. U-net: deep learning for cell counting, detection, and morphometry. *Nature methods*, 16(1):67, 2019.

[31] Özgün Çiçek, Ahmed Abdulkadir, Soeren S Lienkamp, Thomas Brox, and Olaf Ronneberger. 3d u-net: learning dense volumetric segmentation from sparse annotation. In *International conference on medical image computing and computer-assisted intervention*, pages 424–432. Springer, 2016.

[32] Zhengxin Zhang, Qingjie Liu, and Yunhong Wang. Road extraction by deep residual u-net. *IEEE Geoscience and Remote Sensing Letters*, 15(5): 749–753, 2018.

[33] Simon Kohl, Bernardino Romera-Paredes, Clemens Meyer, Jeffrey De Fauw, Joseph R Ledsam, Klaus Maier-Hein, SM Ali Eslami, Danilo Jimenez Rezende, and Olaf Ronneberger. A probabilistic u-net for segmentation of ambiguous images. In *Advances in Neural Information Processing Systems*, pages 6965–6975, 2018.

[34] Gianfranco Fenu, Nikita Jain, Eric Medvet, Felice Andrea Pellegrino, and Myriam Pilutti Namer. On the assessment of segmentation methods for images of mosaics. In *International Conference on Computer Vision Theory and Applications-VISAPP*, volume 3, pages 130–137. SciTePress, SciTePress, 2015.

[35] Diederik P Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2015.

[36] Daniele Panfilo. Le tecnologie per l'anonimizzazione. In *e-Health Data Sharing: Best practices e soluzioni per la condivisione del dato, l'anonimizzazione e la creazione di data lake con dati sanitari.*, pages 35 – 40. Data Valley, 2021.

[37] M. Schulz and J.A. Hennis-Plasschaert. 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, and repealing directive 95/46/ec (general data protection regulation). *Official Journal of the European Union*, 2016.

[38] The European Comission. General data protection regulation (gdpr), 2016. URL `https://gdpr-info.eu/`.

[39] The European Commission. Regulation of the european parliament and of the council: on european data governance (data governance act), 2020.

[40] K. Hänsch and L. Atienza Serna. Directive 95/46/ec of the european parliament and of the council: On the protection of individuals with regard to the processing of personal data and on the free movement of such data. *Official Journal of the European Union*, 1995.

[41] THE EUROPEAN PARLIAMENT AND THE COUNCIL 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, and repealing directive 95/46/ec (general data protection regulation). *Official Journal of the European Union*, 2016.

[42] Center for Medicare and Medicaid Services. Health insurance portability and accountability act, 2022. URL `https://www.cms.gov/regulations-and-guidance/administrative-simplification/hipaa-aca`.

[43] Legal Information Institute. 45 CFR § 160.103, 2022.

[44] The European Comission. Opinion 03/2016 on the evaluation and review of the eprivacy directive (2002/58/ec), 2016. URL `https://ec.europa.eu/justice/article-29/documentation/opinion-recommendation/files/2016/wp240_en.pdf`.

[45] Carl Hempel. Deductive-nomological vs. statistical explanation. In *Scientific Explanation, Space Time (Minnesota Studies in the Philosophy of Science, vol. III)*, pages 98 – 169. Minneapolis: University of Minnesota Press, 1962.

[46] Carl Hempel. Maximal specificity and lawlikeness in probabilistic explanation. *Philosophy of Science*, 35(2):116 – 133, 1968.

[47] James Fretzer. "carl hempel". In *The Stanford Encyclopedia of Philosophy (Spring 2021 Edition), Edward N. Zalta (ed.)*. The Metaphysics Research Lab, Departement of Philosophy, Stanford University, 2021. URL `https://plato.stanford.edu/archives/spr2021/entries/hempel/`.

[48] Article 29 Data Protection Working Party. Opinion 05/2014 on anonymisation techniques, 2014. URL `https://ec.europa.eu/justice/article-29/documentation/opinion-recommendation/files/2014/wp216_en.pdf`.

[49] CMS Law; Tax; Future. GDPR Enforcement tracker, 2022. URL `https://www.enforcementtracker.com/`.

[50] Egress. *Insider Data Breach Survey 2021: Are employees your greatest defense or your biggest vulnerability?* Egress, 2021.

[51] Yves-Alexandre De Montjoye, Laura Radaelli, Vivek Kumar Singh, and Alex "Sandy" Pentland. Unique in the shopping mall: On the reidentifiability of credit card metadata. *Science*, 347(6221):536–539, 2015.

[52] Alex Hern. New york taxi details can be extracted from anonymised data, researchers say. 2014.

[53] Daniel Barth-Jones. The "re-identification" of governor william weld's medical information: a critical re-examination of health data identification risks and privacy protections, then and now. *Then and Now (July 2012)*, 2012.

[54] Arvind Narayanan and Vitaly Shmatikov. How to break anonymity of the netflix prize dataset. *arXiv preprint cs/0610105*, 2006.

[55] Supreme Court of California. RICHARD SANDER et al., Plaintiffs and Appellants, v. STATE BAR OF CALIFORNIA et al., Defendants and Respondents., 2013.

[56] Hao Su, Charles R Qi, Yangyan Li, and Leonidas J Guibas. Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3d model views. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2686–2694, 2015.

[57] C. Dwork and A. Roth. The algorithmic foundations of differential privacy. *Foundations and trends in Theoretical Computer Science*, 9(3-4):211 – 407, 2014.

[58] Hu Li, Jin Li, Guanbiao Lin, Shiyu Peng, Zhenxin Zhang, Yingying Zhang, and Changyu Dong. Defending against Membership Inference Attacks with High Utility by GAN. *IEEE Transactions on Dependable and Secure Computing*, 2022. doi: 10.1109/TDSC.2022.3174569.

[59] Ziqi Zhang, Chao Yan, and Bradley A. Malin. Membership inference attacks against synthetic health data. *Journal of Biomedical Informatics*, 125, 2022.

[60] Steven M. Bellovin, K. Dutta, Preetam, and N. Reitinger. Privacy and synthetic datasets. *Stanford Technology Law Review*, 2018.

[61] Nicholas Carlini, Chang Liu, Úlfar Erlingsson, and Dawn Song. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *Proceedings of the 28th USENIX Security Symposium*, pages 266–284, 2019.

[62] Wojciech Wiewiórowski. *TechSonar: technologies worth monitoring*. European Data Protection Supervisors, 28 September 2021, 2021.

[63] European Comission. *Proposal for a Regulation of the European Parliament and of the Council Laying Down Harmonised Rules on Artificial Intelligence (ARTIFICIAL INTELLIGENCE ACT) and Amending Certain Union Legislative Acts*. European Commission: Brussels, 28 April 2021, 2021.

[64] Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A. Bharath. Generative adversarial networks: An overview. *IEEE Signal Processing Magazine*, 35(1):53–65, 2018. doi: 10.1109/MSP.2017.2765202.

[65] Alankrita Aggarwal, Mamta Mittal, and Gopi Battineni. Generative adversarial networks: An overview of theory and applications. *International Journal of Information Management Data Insights*, 1(1), 2021.

[66] Carl Doersch. Tutorial on variational autoencoders. 2021. doi: 1606.05908v3.

[67] Stefan Hinterstoisser, Olivier Pauly, Hauke Heibel, Marek Martina, and Martin Bokeloh. An annotation saved is an annotation earned: Using fully synthetic training for object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 1–10, 2019.

[68] Subhajit Chatterjee, Debapriya Hazra, Yung-Cheol Byum, and Yong-Woon Kim. Enhancement of Image Classification Using Transfer Learning and GAN-Based Synthetic Data Augmentation. *Mathematics*, 10(1541):1–16, 2022.

[69] Mauro Castelli, Luca Manzoni, Tatiane Espindola, Aleš Popovič, and Andrea De Lorenzo. Generative adversarial networks for generating synthetic features for wi-fi signal quality. *Plos one*, 16(11):e0260308, 2021.

[70] André F. Oliveira, L.F. Juarez, and M.G. Quiles. Molecular Property Prediction and Molecular Design Using a Supervised Grammar Variational Autoencoder. *Journal of Chemical Information and Modeling*, 62(4):817 – 828, 2022.

[71] Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Modeling tabular data using conditional gan. *Advances in Neural Information Processing Systems*, 32:7335–7345, 2019.

[72] Lei Xu and Kalyan Veeramachaneni. Synthesizing tabular data using generative adversarial networks. *arXiv preprint arXiv:1811.11264*, 2018.

[73] Diederik Kingma and Max P. Welling. Auto-encoding variational bayes. *2nd International Conference on Learning Representations, ICLR 2014 - Conference Track Proceedings*, 2014.

[74] Eric Jang, Shixiang Gu, and Ben Poole. Modeling tabular data using conditional gan. *arXiv*, 2019.

[75] B.M. Bolstad, R.A. Irizarry, M. Åstrand, and T.P. Speed. A comparison of normalization methods for high density oligonucleotide array data based on variance and bias. *Bioinformatics*, 19(22):185–193, 2003.

[76] Diane Bouchacourt, Ryota Tomioka, and Sebastian Nowozin. Multi-level variational autoencoder: Learning disentangled representations from grouped observations, 2017.

[77] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.