

## RESEARCH ARTICLE

# A Deep Learning-Based Pipeline for the Generation of Synthetic Tabular Data

DANIELE PANFILO<sup>1,2</sup>, ALEXANDER BOUDEWIJN<sup>2,3</sup>, SEBASTIANO SACCANI<sup>2</sup>, ANDREA COSER<sup>2</sup>, BORUT SVARA<sup>1,2</sup>, CARLO ROSSI CHAUVENET<sup>4</sup>, CIRO ANTONIO MAMI<sup>2,5</sup>, AND ERIC MEDVET<sup>1</sup>, (Member, IEEE)

<sup>1</sup>Department of Engineering and Architecture, University of Trieste, 34127 Trieste, Italy

<sup>2</sup>Aindo s.r.l., 34128 Trieste, Italy

<sup>3</sup>Centre for Industrial Management/Traffic and Infrastructure, Katholieke Universiteit Leuven, 3001 Leuven, Belgium

<sup>4</sup>Angelo Sraffa Department of Legal Studies, Bocconi University, 20100 Milan, Italy

<sup>5</sup>Department of Mathematics and Geosciences, University of Trieste, 34128 Trieste, Italy

Corresponding author: Eric Medvet (emedvet@units.it)

**ABSTRACT** The recent and rapid progresses in Machine Learning (ML) tools and methodologies paved the way for an accessible market of ML services. In principle, small and medium-sized enterprises, as well as big companies, could act as providers and consumers of services, resulting in an intense exchange of ML services where a consumer may ask many providers for a service preview based on its particular business case, that is, its data. In practice, however, many potential service consumers are reluctant to release their data, when seeking for ML services, because of privacy or intellectual property concerns. As a consequence, the market of ML services is not as fluid as it could be. An alternative to providing real data when looking for an ML service consists in generating and releasing synthetic data. The synthetic data should 1) allow the service provider to preview an ML service whose performance is predictive of the one the same service will achieve on the real data; and 2) prevent the disclosure of the real data. In this paper, we propose a data synthesis technique tailored to a family of very relevant business cases: supervised and unsupervised learning on single-table datasets and relational datasets. Our technique is based on generative deep learning models and we instantiate it in three variants: standard Variational Autoencoders (VAEs),  $\beta$ -VAEs, and Introspective VAEs. We experimentally evaluate the two variants to measure the degree to which they meet the two requirements above, using several performance indexes that capture different aspects of the quality of the generated data. The results suggest that data synthesis is a practical answer to the need of decoupling ML service providers and consumers and, hence, can favor the arising of an active and accessible market of ML services.


**INDEX TERMS** Synthetic data, variational auto encoders, data privacy, tabular data.

## I. INTRODUCTION

An essential part of the European Commission (EC) digital strategy is titled “Shaping Europe’s Digital Future” [2]. It aims at setting the worldwide standard for horizontal and vertical data sharing through the “Towards Common Data Spaces” initiative [3]. 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

The associate editor coordinating the review of this manuscript and approving it for publication was Rahim Rahmani .

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.

Data security can be characterized through three risk factors [4]: (a) the risk of identifying individual records; (b) the risk of linking records to within the dataset or across datasets; (c) the risk of inferring confidential parameter values in a data set. *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 susceptible to all three risks [5]. Moreover, the degree of anonymization directly affects the quality of the information still contained in the dataset.

The risk of re-identification in anonymized datasets is not just a theoretical concern. De Montjoye et al. [6] showed that four data points (location and time of the purchase using a credit card) are enough to uniquely re-identify 90% of individuals. Indeed, many anecdotal evidences appear to confirm the issues of anonymization with respect to re-identification. In 1997, the Massachusetts Governor's medical records were identified by matching anonymous data from medical meetings with publicly available voter registration data [7]. 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 [8], forcing Netflix to remove the data mere days after the publication. A customer sued for breach of privacy; Netflix settled. These cases stress the need for a methodology to securely obtain reliable analytic results without the risk of re-identification.

Deep learning (DL)-based *generative models* provide a reliable alternative to data anonymization [9]. 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., [10], [11]). 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 [12] (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 paper, we propose a pipeline for generating synthetic data for the kind of business scenario described above. Our contribution is twofold. First, we formally state the problem of synthetic data generation for the case of tabular data (possibly relational) with heterogeneous data types and missing values. In this problem statement, we also define a set of performance indexes that are suitable to assess any solution (i.e., any synthetic data generation technique for tabular data) 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*). For the former axis, we consider the concrete case in which a ML service provider builds an ML system based on the synthetic data and the client can use the quality of the built system as a predictor for the quality of the same kind of system built on the original data. As performance indexes, we consider both intrinsic (i.e., evaluated using merely properties of original and synthetic data sets) and extrinsic (i.e., evaluated through actual ML systems) measures.

Second, we design, describe in detail, and experimentally evaluate 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. Our pipeline is based on Variational Autoencoders (VAEs), that we exploit in three variants: vanilla VAEs,  $\beta$ -VAE, and introspective VAEs. Central to this approach is the introduction of a novel data pre-processing methodology that we designed to accurately model missing data values and temporal data attributes with seasonality. We experimentally evaluate our pipeline on three publicly available datasets from the UCI Machine Learning Repository [1] and one custom relational dataset. Results indicate that our approach is able to generate synthetic data with strong utility-preservation and disclosure-averseness.

In this work, we chose VAE-based models for synthetic data generation. Alternatively, generative adversarial methods (GANs) could have been chosen. However, training this latter model 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. Reliance on random initially sampled points hinder the robustness of the output for tabular 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 to identify relationships that are causal rather than merely correlations.

The remainder of this paper is organized as follows. In Section II we survey relevant previous works that dealt with the task of generating synthetic data (Section II-A) and we provide the legal framework for data privacy preservation (Section II-B). In Section III-A we give the formal problem statement for synthetic data generation for tabular data with heterogeneous data types and missing values; in this part of the paper, in Sections III-C1 and III-C2, we also define the performance indexes we propose for assessing any solution to the problem. In Section IV we describe in detail our approach, including the data pre-processing steps (Section IV-A) and the way we use VAEs to generate synthetic data (Section IV-B). In Section V we present the experimental evaluation we performed to assess our approach and discuss the results. Finally, in Section VI, we draw the conclusions.

## II. RELATED WORKS

### A. DL FOR DATA SYNTHESIS

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. [13] 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 [14]. Park et al. [12] 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. [15], the authors combine GANs with transfer learning to augment datasets to better train ML algorithms for image classification. Castelli et al. [16] 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. [17] 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. [18] propose a conditional GAN named CTGAN

where a conditional generator is used to synthesize eight different datasets. Xu and Veeramachaneni [19] employ both a GAN variation named Tabular GAN (TGAN) to address the same problem. Li et al. [20] 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.

### B. LEGAL FRAMEWORK FOR DATA PRIVACY

The General Data Protection Act (GDPR [21], [22]) 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” [2]. At the same time, the “Towards Common Data Spaces” initiative will facilitate enhanced data exchange [3]. An important part of the initiative is the Data Governance Act, adopted in 2021 [23]. 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” ([22]).

Neither the GDPR nor its predecessor, Directive 95/46/EC [24], clarify when data can be considered sufficiently anonymous. A working definition is, however, provided in Article 4 of Regulation EU 2016679 [25]. 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. The directive does not further specify indirect identification processes.

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 [4]). In Opinion 03/2016 of the

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

**Re-identification attack**, in which the attacker identifies an individual in the (anonymized) dataset.

**Linkage attack**, in which the attacker can link 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.

**Parameter inference attack**, in which the attacker can deduce the value of a parameter from the data.

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. Processed data must then not contain any link between the information and the original data subject. 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 classifies data anonymization techniques common in practice into three categories: randomization, generalization, and pseudonymization.

*Randomization* is achieved through adding noise to the data ("noise addition"; "differential privacy") or by permuting attribute values ("permutation"). Noise addition is a collection of methods that alter attribute values. 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. Noise addition techniques add noise once to the data, prior to its exchange. Differential privacy is a related technique, in which the noise is added ad hoc when data is queried. 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.

*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. The most common generalization method is  $k$ -anonymity, in which attributes are generalized to the point where at least  $k$  individuals share each value. Note that  $k$ -anonymity results in a loss of information, as the specific values are removed. 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. This further reduces the risk of re-identification and linkage attacks. In the previous example, knowing an individual's country of birth is now insufficient for a

re-identification attack. Not only are there at least  $k$  individuals with the same country of birth, there are also at least  $l$  highly distinctive records associated with them. As the combination of  $k$ -anonymity and  $l$ -diversity may distort the original data set, an additional constraint,  $t$ -closeness, is introduced. This requires that each value is present as many times as needed to mimic the initial distribution of each attribute.

*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.

Noise addition-anonymized data was shown susceptible to re-identification and linkage attacks in practical cases [8], [13]. It offers no protection against parameter inference attacks and is insufficient as a standalone anonymization method [4]. Similar arguments can be made about differential privacy. Likewise, in datasets anonymized through permutation, correlations between parameters make inverting the permutation operations possible. As such, all three attacks can still be executed successfully [4].

Likewise, generalization techniques remove a lot of the information contained in a dataset, yet offer limited protection. While the risks of re-identification and linkage attacks is reduced after generalization, parameter inference attacks can still be executed successfully. This is because correlations between parameters are still contained in the data. These can be exploited to infer properties of the generalized classes of subjects.

Data obtained after applying pseudonymization cannot be considered fully anonymized [4]. Individuals can easily be re-identified using values of other attributes. These values also make linkage attacks highly applicable. Non-pseudonymized attributes are not even altered, hence mere inspection of the data can already be considered an attribute inference attack. The methods should therefore be considered merely an additional layer of protection.

In conclusion, the traditional anonymization methods insufficiently protect against the three types of attacks. As such, these methods are not in line with the GDPR regulation, explicitly demanding such protection. Furthermore, the degree of protection these methods provide is negatively correlated with the degree to which they dismantle the original data. For instance, data anonymized using noise addition can only be considered anonymized if the impact of the noise outweighs the information contained in the dataset [4]. When combining multiple anonymization methods, the quality of the information diminishes even further.

*Synthetic data* generated using deep generative models form a new paradigm in secure data transfer. Every single record 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 may

not be easily attributed to a single data subject (e.g., very specific location data). 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. In consequence, synthetic data is more defensible from a legal perspective.

### III. 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.

#### A. 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  $v(t, a) \in V(T, a) \cup \emptyset$  the value of the attribute  $a$  for the row  $t$ , with  $V(T, a)$  being the *domain* of the attribute  $a$  and  $\emptyset$  representing the undefined value (i.e.,  $v(t, a) = \emptyset$  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*  $T^*$  has a unique attribute  $a^*$ ; (c) for each other *secondary table*  $T$  in the dataset,  $A(T) \ni a^*$  and  $\forall t \in T, \exists t^* \in T^* : v(t, a^*) = v(t^*, a^*)$ . Intuitively, a relational dataset is a dataset with a primary table  $T^*$  describing some entities, one row per entity, and other tables describing some other entities, each one linked to one specific entity of  $T^*$ . 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  $a^*$  of the primary table  $T^*$  is of type *discrete* and its values are  $1, 2, \dots \in \mathbb{N}$ , i.e., for each  $t_i \in T^*, v(t_i, a^*) = i$ .

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  $\phi : D \rightarrow D'$  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  $T_{\text{user}}^*$  describing users, with the  $|A(T_{\text{user}}^*)| = 4$  attributes `id*`, `age`, `country`, `gender`, `id*` being unique, and a table  $T_{\text{purchases}}$  describing purchases made by users, with the  $|A(T_{\text{purchases}})| = 5$  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  $V(\text{prodPrice}, T_{\text{purchases}}) = \mathbb{R}^+$ ).

#### B. ML-BASED SOLUTION AND EFFECTIVENESS METRIC

The use cases we envision for synthetic data are supervised learning problems, that include the very practically relevant problems of classification and regression.

More formally, given a dataset  $D$  with at least one table  $T$  with a *target attribute*  $\hat{a}$ , a *problem* consists in finding a way for predicting the value  $v(t, \hat{a})$  of  $\hat{a}$  for a row  $t$  for which the values of the other attributes  $\{a_1, a_2, \dots\} = A(T) \setminus \{\hat{a}\}$  are known, i.e., in finding a *model*  $m : V(a_1, T) \times V(a_2, T) \times \dots \rightarrow V(\hat{a}, T)$  such that  $m(t) \approx v(t, \hat{a})$ .

Given a problem, i.e., a triplet  $D, T, \hat{a}$ , a ML-based *solution* is a procedure that, given a *learning dataset*  $D'$  compatible with  $D$ , outputs a model  $m$  for the problem.

An *effectiveness metric* is a procedure that, given a model  $m$  and a *testing dataset*  $D_{\text{test}}$  compatible with the one on which the model has been built, outputs a numerical value  $e(m, D_{\text{test}})$  whose semantic is, without loss of generality, the greater, the more useful the model for solving the problem.

As an example consider the relational dataset  $D$  composed of the tables  $T_{\text{user}}^*$  and  $T_{\text{purchases}}$  described above. A problem might be the one of predicting  $\hat{a} = \text{gender}$ . A possible solution would be the one of learning a model with Random Forest using `age` and `country` as features. An effectiveness metric could be the accuracy of the model on the testing dataset. A likely better solution might involve some processing of the other table  $T_{\text{purchases}}$  (i.e., its corresponding table in

the learning dataset): for this solution, the accuracy might be greater.

### C. SYNTHETIC DATA GENERATION

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 paper we introduce a few ways for measuring *utility-preserving* and *disclosure-averse* abilities of a synthetic dataset. We present them in the following sections.

#### 1) UTILITY-PRESERVATION (UP)

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  $D$ .

##### a: 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:

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

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

In practice,  $MC(\hat{D}, D)$  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  $MC(\hat{D}, D)$  to 0. That is, the more similar  $\hat{D}$  and  $D$ .

In the experiments presented in Section V, we use MC by choosing a few problems and a few solutions in order to obtain an aggregate measure that averages possible lucky and unlucky conditions.

##### b: 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  $\phi$  between their tables, we define the PCD of  $\hat{D}$  and  $D$  as:

$$PCD(\hat{D}, D) = \frac{1}{|D|} \sum_{T \in D} \frac{1}{\alpha_T} \|C(T) - C(\phi(T))\|_F \quad (2)$$

where  $|D|$  is the number of tables in  $D$  (and hence  $\hat{D}$ ) and  $C(T)$  is the correlation matrix of attributes in  $T$ ,  $\alpha_T$  is a normalization factor, and  $\|\cdot\|_F$  is the Frobenius norm of matrices, i.e.,  $\|A\|_F = \sqrt{\sum_i \sum_j |a_{i,j}|^2}$ . 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  $\alpha_T = \sqrt{4(|A(T)|^2 - |A(T)|)}$ , 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,  $PCD(\hat{D}, D)$  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  $PCD(\hat{D}, D)$  to 0. That is, the more similar  $\hat{D}$  and  $D$ .

##### c: 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:

$$CSE(\hat{D}, D) = \frac{\alpha}{k} \sum_{i=1}^{i=k} \left| \frac{|C_i \cap \hat{T}|}{|C_i|} - \frac{|\hat{T}|}{|T| + |\hat{T}|} \right| \quad (3)$$

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  $\alpha$  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  $\alpha$  to:

$$\alpha = \begin{cases} \frac{|T| + |\hat{T}|}{|T|} & \text{if } |\hat{T}| \leq |T| \\ \frac{|T| + |\hat{T}|}{|\hat{T}|} & \text{otherwise} \end{cases} \quad (4)$$

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.

In the experiments presented in Section V, we use  $k$ -means as clustering technique, choosing the value of  $k$  automatically in each case with the elbow method.  $k$ -means requires that all the attributes in the table are numeric. In the later sections, we will show how we can transform any dataset in a single-table dataset in which all attributes are numeric: by applying this transformation to both  $D$  and  $\hat{D}$ , we can measure CSE for any pair of compatible datasets.

## 2) 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.

### $\alpha$ : NEAREST NEIGHBOR DISTANCE RATIO (NNDR $_{\mu}$ , NNDR $_{\sigma}$ , AND NNDR $_{\rho}$ )

The three intrinsic measures of DA are all based on 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:

$$\begin{aligned} \text{NNDR}_{\mu}(\hat{D}, D) &= \left| 1 - \frac{1}{|T|} \sum_{t \in T} \frac{\min_{\hat{t} \in \hat{T}} d(t, \hat{t})}{\min_{t' \in T \setminus \{t\}} d(t, t')} \right| \\ &= \left| 1 - \frac{1}{|T|} \sum_{t \in T} \rho(t) \right| \end{aligned} \quad (5)$$

In practice, NNDR $_{\mu}$  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 $_{\mu}$  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:

$$\text{NNDR}_{\sigma}(\hat{D}, D) = \sqrt{\frac{\sum_{t \in T} \left( \rho(t) - \frac{1}{|T|} \sum_{t \in T} \rho(t) \right)^2}{|T| - 1}} \quad (6)$$

In practice, the lower NNDR $_{\sigma}$ , the more even the value of  $\rho$  across entities, i.e., the harder to identify some of them (for the same value of NNDR $_{\mu}$ ). Moreover, if NNDR $_{\sigma}$  is large and NNDR $_{\mu}$  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:

$$\text{NNDR}_{\rho}(\hat{D}, D) = \left| \frac{1}{2} - \frac{1}{|T|} |\{t \in T : \rho(t) < 1\}| \right| \quad (7)$$

In practice, NNDR $_{\rho}$  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 $_{\rho}$ , the lower the proportion of original rows that are closer to synthetic rows than to other original rows.

### $b$ : NEAREST NEIGHBOR DISTRIBUTIONS DIFFERENCE (NNDD)

Finally, we define the nearest neighbor distributions difference as follows. Given the distributions of the values of the distance to the closest original and synthetic neighbors across original rows  $t \in T$ , NNDD( $\hat{D}, D$ ) 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  $\alpha$ .

## IV. OUR APPROACH

We propose to solve the problem stated in Section III-C with a data synthesis pipeline whose core is based on variational autoencoders (VAEs). Since VAEs operate on numerical data, the first and last steps of our pipeline take care of converting all the attribute values in the actual dataset  $D$  to numbers (in the first step) and converting the numbers generated by the VAE back to the suitable data types (in the last step) to populate the synthetic dataset  $\hat{D}$ . Moreover, we include in the

pipeline a special treatment of relational datasets that allows to (a) restructure them in such a way that they can be fed to a VAE and (b) take into account the number of rows in each secondary table that are associated with the respective entity in the primary table.

In the following sections, we describe the pipeline in detail.

### A. CONVERSION TO AND FROM NUMBERS

In general, we convert values of attributes whose type is one of the five types described in Section III-A 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  $\phi_{\text{type}} : V(T, a) \rightarrow \mathbb{R}^p$  and  $\phi_{\text{type}}^{-1} : \mathbb{R}^p \rightarrow V(T, a)$ . We describe how we build  $\phi_{\text{type}}$  and  $\phi_{\text{type}}^{-1}$ , 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  $X = \{\mathbf{x}_i\}_i$  of numerical vectors, one vector per row, resulting from the concatenation of the single vectors obtained by converting each attribute:

$$\begin{aligned} \mathbf{x}_i &= [\mathbf{x}_{i,a_1} \ \mathbf{x}_{i,a_2} \ \dots] \\ &= [\phi_{\text{type}}(v(t_i, a_1)) \ \phi_{\text{type}}(v(t_i, a_1)) \ \dots] \end{aligned} \quad (8)$$

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  $\mathbf{x}_i$  in chunks of proper size, then use the corresponding  $\phi_{\text{type}}^{-1}$  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  $T := \phi_{\text{type}}^{-1}(X)$ .

#### 1) 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 [26].

Formally, let  $\{v(t_i, a)\}_i$  be the defined values (i.e., not  $\emptyset$ ) 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  $\phi_{\text{type}} : [q_1, q_k] \rightarrow [0, 1]$  as:

$$\phi_{\text{type}}(v) = \begin{cases} \frac{j}{k} & \text{if } v \neq \emptyset \\ 0.5 & \text{otherwise} \end{cases} \quad (9)$$

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

$$\phi_{\text{type}}^{-1}(x) = q_j + (q_{j+1} - q_j)(kx - j) \quad (10)$$

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.

#### 2) 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  $\mathbf{b}$  with  $\frac{1}{\delta\tau}(\tau_{\max} - \tau_{\min})$  elements, where  $\delta\tau$  is a parameter of the conversion, as:

$$b_j = |\{i : \tau_{\min} + j\delta\tau \leq v(t_i, a) < \tau_{\min} + (j+1)\delta\tau\}| \quad (11)$$

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  $\mathbf{b}$  and obtain the period  $\tau^*$  of the strongest harmonic of  $\mathbf{b}$ . Then, we define  $\phi_{\text{type}} : [\tau_{\min}, \tau_{\max}] \rightarrow [0, 1]^2$  as:

$$\phi_{\text{type}}(v) = \begin{cases} (x_1, x_2) & \text{if } v \neq \emptyset \\ (0, 5, 0.5) & \text{otherwise} \end{cases} \quad (12)$$

with:

$$x_1 = \left\lfloor \frac{v - \tau_{\min}}{\tau^*} \right\rfloor \frac{\tau^*}{\tau_{\max} - \tau_{\min}} \quad (13)$$

$$x_2 = \frac{v \bmod \tau^*}{\tau^*} \quad (14)$$

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

$$\phi_{\text{type}}^{-1}(x) = \phi_{\text{type}}^{-1}(x_1, x_2) = \tau_{\min} + x_1(\tau_{\max} - \tau_{\min}) + x_2\tau^* \quad (15)$$

In practice, given the periodicity  $\tau^*$  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.



### 3) CATEGORICAL AND BINARY ATTRIBUTES

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

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

$$\phi_{\text{type}}(v) = \begin{cases} \mathbf{x} & \text{if } v \neq \emptyset \\ (0.5, \dots, 0.5) & \text{otherwise} \end{cases} \quad (16)$$

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

$$x_i = \begin{cases} 1 & \text{if } v = c_i \\ 0 & \text{otherwise} \end{cases} \quad (17)$$

We define  $\phi_{\text{type}}^{-1} : [0, 1]^{|C|} \rightarrow C$  as:

$$\phi_{\text{type}}^{-1}(\mathbf{x}) = c_j \quad (18)$$

with  $j = \arg \max_i x_i$ .

### 4) 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  $\emptyset$ .

Formally, for each attribute  $a$  and each table  $T$  such that for at least one row  $t_i \in T$  the value  $v(t_i, T) = \emptyset$ , 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.,  $\emptyset$ ). We define this further mapping based on the previously described cases:

$$\phi'_{\text{type}}(v) = \begin{cases} [\phi_{\text{type}}(v) \ 0] & \text{if } v \neq \emptyset \\ [\phi_{\text{type}}(v) \ 1] & \text{otherwise} \end{cases} \quad (19)$$

and:

$$\phi'^{-1}_{\text{type}}(\mathbf{x}) = \begin{cases} \phi_{\text{type}}^{-1}(v) & \text{if } x_{\text{last}} > 0.5 \\ \emptyset & \text{otherwise} \end{cases} \quad (20)$$

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

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  $\emptyset$ .

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.

## B. USING VAEs for GENERATING SYNTHETIC NUMERICAL DATA

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

In the following sections, we describe steps 2, 3, and 4 in detail. We described steps 1 and 5 in the previous sections.

### 1) VARIATIONAL AUTOENCODERS (VAEs) BACKGROUND

We considered three variants of VAEs for generating numerical data: standard variational autoencoders (VAE), beta-variational autoencoders ( $\beta$ -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 [27]. 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.

$\beta$ -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.  $\beta$ -VAEs are promising in synthetic 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  $\beta$  is used to manage their relative importance [10].

IntroVAEs form a hybrid framework that combine VAE with generative adversarial networks (GANs) [28]. 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 [11]. While VAE and  $\beta$ -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 ( $\beta$ -)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 [28]. The use of this loss function to make IntroVAE applicable to tabular data is a novel contribution of our work.

## 2) LEARNING A VAE

From the point of view of our approach, all the three variants (standard VAE,  $\beta$ -VAE, and IntroVAE) can be seen as “black boxes” that given a dataset  $X = \{\mathbf{x}_i\}_i$ , with  $\mathbf{x}_i \in \mathbb{R}^q$ , 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  $\phi_{\text{enc}} : \mathbb{R}^q \rightarrow \mathbb{R}^r$  and  $\phi_{\text{dec}} : \mathbb{R}^r \rightarrow \mathbb{R}^q$  (encoder and decoder, respectively). The two functions are neural networks and are completely described by a vector  $\theta = [\theta_{\text{enc}} \theta_{\text{dec}}] \in \mathbb{R}^m$  of numerical parameters, whose size  $m$  depends on the variant. We obtain  $\theta$  from  $X$  in a learning process which is a gradient-based optimization driven by a loss function  $L : \mathbb{R}^m \rightarrow \mathbb{R}$  such that the lower  $L(\theta)$ , the better the pair  $\phi_{\text{enc}}, \phi_{\text{dec}}$  described by  $\theta$ .

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  $\phi_{\text{dec}}$  defines a

probability distribution depending on parameters  $\theta_{\text{dec}}$ , such that  $p(\mathbf{x}|z, \theta_{\text{dec}})$  is the probability that  $\mathbf{x}$  was observed, given that the latent variables were  $z$  and the model parameters were  $\theta_{\text{dec}}$ . Symmetrically, the encoder  $\phi_{\text{enc}}$  defines the distribution of the latent space for given observations parametrized by  $\theta_{\text{enc}}$ . Thus,  $p(z|\mathbf{x}, \theta_{\text{enc}})$  is the probability that  $z$  was the latent vector, given that  $\mathbf{x}$  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:

$$\begin{aligned} L(\theta) &= \mathbb{E}_{z \sim p(z|\mathbf{x}, \theta_{\text{enc}})} \log p(\mathbf{x}|z, \theta_{\text{dec}}) \\ &= \mathbb{E}_{\phi_{\text{enc}}} \log \phi_{\text{dec}} \end{aligned} \quad (21)$$

where the latter is a shorthand for the former.

For  $\beta$ -VAEs, a constraint is imposed that the encoder  $\phi_{\text{enc}}$  must also be sufficiently similar to a given prior  $P(z)$  (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 is also invoked to achieve this. The constraint is incorporated into the loss function through the KKT-conditions. Hence,  $\beta$ -VAEs we use the following loss function:

$$L(\theta) = -\mathbb{E}_{\phi_{\text{enc}}} \log \phi_{\text{dec}} + \beta D_{\text{KL}}(\phi_{\text{enc}} \parallel \mathcal{N}) \quad (22)$$

where  $D_{\text{KL}}(\phi_{\text{enc}} \parallel \mathcal{N}(\mathbf{0}, \mathbf{1}))$  denotes the KL-divergence between distributions  $p(z|\mathbf{x}, \theta_{\text{enc}})$  and  $\mathcal{N}(\mathbf{0}, \mathbf{1})$ . The parameter  $\beta$  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 [11], that we do not report here for brevity.

Concerning the optimizer, we use ADAM with standards parametrization.

## 3) LATENT DATA GENERATION AND MAPPING

Given a target size  $n$  of the synthetic data  $n = |\hat{X}|$ , we generate the collection  $\hat{Y} = \{\hat{y}_j\}_j$  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  $Y = \{y_i\}_i$ , with  $y_i = \phi_{\text{enc}}(\mathbf{x}_i) \in \mathbb{R}^r$ ; we denote this step as  $Y := \phi_{\text{enc}}(X)$ . Then, we fit a multivariate normal distribution over  $Y$  and obtain its parameters  $\mu \in \mathbb{R}^r$  and  $\Sigma \in \mathbb{R}^{r \times r}$ ; we denote this step as  $(\mu, \Sigma) := \mathcal{N}^{-1}(Y)$ . Finally, we sample the multivariate normal distribution  $\mathcal{N}(\mu, \Sigma)$   $n$  times and obtain  $\hat{Y}$ ; we denote this step as  $\hat{Y} \stackrel{\sim}{\sim} \mathcal{N}(\mu, \Sigma)$ . As the last step, we obtain  $\hat{X} := \{\hat{x}_j\}_j$  by applying the decoder to  $\hat{Y}$ , i.e.,  $\forall j, \hat{x}_j := \phi_{\text{dec}}(\hat{y}_j)$ ; we denote this step as  $\hat{X} = \phi_{\text{dec}}(\hat{Y})$ .

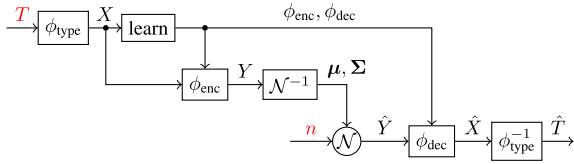
Summarizing, the entire process from  $T$  to  $\hat{T}$  consists of the following steps:

$$X := \phi_{\text{type}}(T) \quad (23)$$

$$(\phi_{\text{enc}}, \phi_{\text{dec}}) := \text{learn}(X) \quad (24)$$

$$Y := \phi_{\text{enc}}(X) \quad (25)$$

$$(\mu, \Sigma) := \mathcal{N}^{-1}(Y) \quad (26)$$



**FIGURE 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.

$$\hat{Y} \stackrel{n}{\sim} \mathcal{N}(\mu, \Sigma) \quad (27)$$

$$\hat{X} := \phi_{\text{dec}}(\hat{Y}) \quad (28)$$

$$\hat{T} := \phi_{\text{type}}^{-1}(\hat{X}) \quad (29)$$

The same steps are depicted in Figure 1.

### C. 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 [29]. 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 [29] 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 IV-B 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  $T^*$  and by  $h$  secondary tables  $T^1, \dots, T^h$ . We proceed as follows.

#### 1) PRIMARY TABLE

We pre-process the primary table  $T^*$ , obtaining a table  $T'^*$ , by adding one new discrete attribute  $a_{T^l}$  for each secondary table  $T^l$ . For each row  $t$  in  $T^*$  and each  $l$ , we set the value  $v(t, a_{T^l})$  to the number of rows  $t'$  in  $T^l$  such that  $v(t', a^*) = v(t, a^*)$ . 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  $a^*$  from  $A(T^*)$ .

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

We build  $Y^*, \hat{Y}^*$ , and  $\hat{X}^*$  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  $\hat{X}^*$ . Then, when building the synthetic primary table, we apply  $\phi_{\text{type}}^{-1}$  to  $\hat{X}^*$ , obtaining  $\hat{T}'^* = \phi_{\text{type}}^{-1}(\hat{X}^*)$ , and finally post-process  $\hat{T}'^*$ , obtaining  $\hat{T}^*$ , by removing the attributes  $a_{T^1}, \dots, a_{T^h}$  and by adding again  $a^*$ , whose values are set to  $v(t_j, a^*) = j$ .

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

$$T'^* := \text{preProc}(T^*) \quad (30)$$

$$X^* := \phi_{\text{type}}^*(T'^*) \quad (31)$$

$$(\phi_{\text{enc}}^*, \phi_{\text{dec}}^*) := \text{learn}(X^*) \quad (32)$$

$$Y^* := \phi_{\text{enc}}^*(X^*) \quad (33)$$

$$(\mu^*, \Sigma^*) := \mathcal{N}^{-1}(Y^*) \quad (34)$$

$$\hat{Y}^* \stackrel{n}{\sim} \mathcal{N}(\mu^*, \Sigma^*) \quad (35)$$

$$\hat{X}^* := \phi_{\text{dec}}^*(\hat{Y}^*) \quad (36)$$

$$\hat{T}'^* := \phi_{\text{type}}^{-1}(\hat{X}^*) \quad (37)$$

$$\hat{T}^* := \text{postProc}(\hat{T}'^*) \quad (38)$$

#### 2) 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  $a^*$ , 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  $\phi_{\text{enc}}^l, \phi_{\text{dec}}^l$  from  $X^l$ . In this ML-VAE, the encoder is a function  $\phi_{\text{enc}}^l : \mathbb{R}^{q^l} \rightarrow \mathbb{R}^{r^l}$ ,  $q^l$  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  $\phi_{\text{dec}}^l : \mathbb{R}^{r^l+r^*} \rightarrow \mathbb{R}^{q^l}$ : 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  $r^*$  more variables than the latter. When learning the ML-VAE on  $X^l$ , i.e., when computing the loss for elements  $\{x_i^l\}_i$  of  $X^l$ , each point in the latent space is obtained as a row-wise concatenation:

$$y_i^l = [\underbrace{\phi_{\text{enc}}^l(x_i^l)}_{\in \mathbb{R}^{r^l}} \underbrace{\phi_{\text{enc}}^*(x^*)}_{\in \mathbb{R}^{r^*}}] \in \mathbb{R}^{r^l+r^*} \quad (39)$$

where  $x^*$  is the element of  $X^*$  corresponding to the row  $t^*$  of the primary table such that  $v(t^*, a^*) = v(t_i^l, a^*)$ , i.e., the

primary row related to the row  $x_i^l$  of the secondary table  $T^l$ . As a shorthand, we write collectively that:

$$Y^l = \{y_i^l\}_i = \left\{ [\phi_{\text{enc}}^l(x_i^l) \phi_{\text{enc}}^*(x^*)]_i \right\} \\ = \left[ \phi_{\text{enc}}^l(X^l) \phi_{\text{enc}}^*(\text{prim}(X^l)) \right] \quad (40)$$

When learning this ML-VAE, we use the same optimizer described in Section IV-B2 and the following loss function:

$$L(\theta) = \frac{1}{|T^*|} \sum_{l=1}^{|D|-1} \sum_{v^* \in V(T^*, a^*)} (L_{\text{reg}}(l, v^*, \theta) - \beta_l L_{\text{KL}}(l, v^*, \theta)) \quad (41)$$

where:

$$L_{\text{reg}}(l, v^*, \theta) = \sum_{t_i^l: v(t_i^l, a^*)=v^*} \mathbb{E}_{\phi_{\text{enc}}^l(v^*)} \mathbb{E}_{\phi_{\text{enc}}^l(t_i^l)} \log \phi_{\text{dec}}^l \quad (42)$$

$$L_{\text{KL}}(l, v^*, \theta) = \sum_{t_i^l: v(t_i^l, a^*)=v^*} D_{\text{KL}}(\phi_{\text{enc}}^l(t_i^l) \| P_{(l, v^*)}) \quad (43)$$

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

Then, we extract the rounded values  $n_1^l, \dots, n_n^l$  of the attribute  $a_{T^l}$  from  $\hat{T}^*$ , with  $n = |\hat{T}^*|$ . Each  $n_k^l$  represents the number of rows in the synthetic secondary table  $\hat{T}^l$  that should refer to the  $k$ -th primary entity in  $\hat{T}^*$ . We also take the corresponding points  $\hat{y}_1^*, \dots, \hat{y}_n^*$  in the primary latent space.

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

- (1) compute  $Y^{l,k}$  as the image  $\phi_{\text{enc}}^l(X^{l,k})$  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.,  $X^{l,k} = \{x_i^l : v(t_i^l, a^*) = k\}$ ;
- (2) fit a multivariate normal distribution over  $Y^{l,k}$  obtaining  $\mu^{l,k}$  and  $\Sigma^{l,k}$ ;
- (3) sample the multivariate normal distribution  $\mathcal{N}(\mu^{l,k}, \Sigma^{l,k})$   $n_k^l$  times obtaining  $\hat{Y}^{l,k}$ ;
- (4) build  $\hat{Y}^{l,k}$  by row-wise concatenation of  $\hat{Y}^{l,k}$  and  $n_k^l$  copies of  $\hat{y}_k^*$ , that constitute the disentangled variables representing the  $k$ -th primary entity;
- (5) compute  $\hat{X}^{l,k}$  as the image of  $\phi_{\text{enc}}^l(\hat{Y}^{l,k})$ .

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.,  $\hat{X}^l = \bigcup_{k=1}^n \hat{X}^{l,k}$ . Finally, we obtain  $\hat{T}^{l,l}$  by applying  $\phi_{\text{type}}^{-1}$  on  $\hat{X}^l$  and then post-process  $\hat{T}^{l,l}$  by adding the attribute  $a^*$ : 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  $\hat{T}^l$ :

$$T^{l,l} := \text{preProc}(T^l) \quad (44)$$

$$X^l := \phi_{\text{type}}^l(T^{l,l}) \quad (45)$$

$$(\phi_{\text{enc}}^l, \phi_{\text{dec}}^l) := \text{learn}(X^l) \quad (46)$$

$$X^{l,k} := \{x_i^l : v(t_i^l, a^*) = k\} \quad \forall k \quad (47)$$

$$Y^{l,k} := \phi_{\text{enc}}^l(X^{l,k}) \quad \forall k \quad (48)$$

$$(\mu^{l,k}, \Sigma^{l,k}) := \mathcal{N}^{-1}(Y^{l,k}) \quad \forall k \quad (49)$$

$$\hat{Y}^{l,k} \sim \mathcal{N}(\mu^{l,k}, \Sigma^{l,k}) \quad \forall k \quad (50)$$

$$\hat{Y}^{l,k} := [\hat{Y}^{l,k} \text{ nOf}(\hat{y}_k^*, n_k^l)] \quad \forall k \quad (51)$$

$$\hat{X}^{l,k} := \phi_{\text{dec}}^l(\hat{Y}^{l,k}) \quad \forall k \quad (52)$$

$$\hat{X}^l := \bigcup_{k=1}^n \hat{X}^{l,k} \quad (53)$$

$$\hat{T}^{l,l} := \phi_{\text{type}}^{-1}(\hat{X}^l) \quad (54)$$

$$\hat{T}^l := \text{postProc}(\hat{T}^{l,l}) \quad (55)$$

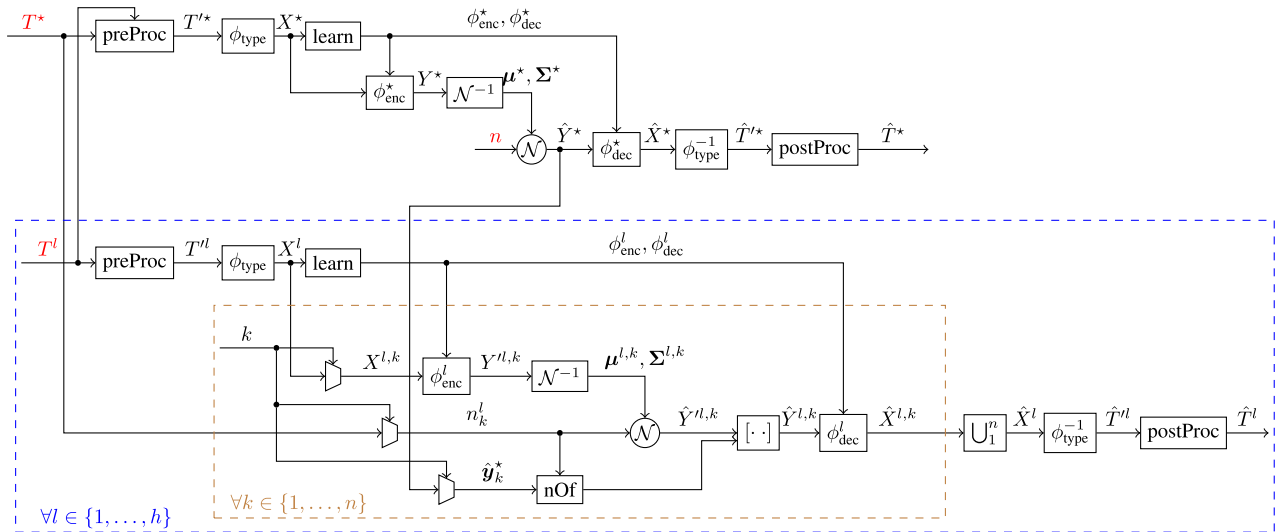
We remark that, for secondary tables, the number of rows in the table is not chosen by the user, but is instead generated as part of the synthetic data. Hence, we meet the first goal stated in Section IV-C. Moreover, since we reserve a portion of the latent space for disentangled variables that represents the primary entity, and since we use several copies of the same values, we meet the second goal stated in Section IV-C: 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 2.

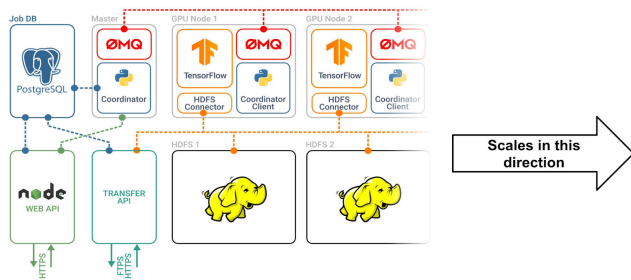
#### D. IMPLEMENTATION

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  $\theta$  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



**FIGURE 2.** Schematic representation of the steps for going from a relational dataset  $D = (T^*, T^1, \dots, T^h)$  to a compatible synthetic relational dataset  $\hat{D} = (\hat{T}^*, \hat{T}^1, \dots, \hat{T}^h)$ . In red, the parameter and inputs provided by the user; in colored dashed rectangles, parts of the steps that are repeated for  $l$  and  $k$ .



**FIGURE 3.** Overview of the implementation as a SaaS.

training, compute nodes generate the requested amount of synthetic data and store them to HDFS. The coordinator also takes care of training and generating jobs and of the resource allocation for each of them, using a message queue. The architecture is depicted in Figure 3.

**V. EXPERIMENTAL EVALUATION**

We performed an experimental campaign aimed at assessing to which degree our data synthesis pipeline meet the two requirements stated in Section III. To this end, we considered 4 datasets, one of which being relational, applied our technique several times, and measured the performance indexes defined in Section III-C1 (for UP) and Section III-C2 (for DA) on the generated synthetic datasets.

**A. DATASETS**

We took three single-table datasets from the UCI Machine Learning Repository [1]: Insurance, Covertypes, Bank. The size of the corresponding tables ranges from  $\approx 1000$  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

**TABLE 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.

Dataset	$ T $	$ A(T) $	R+D	B	C	T	Target
Insurance	1338	14	14	0	0	0	R
Covertypes	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	

a predefined target attribute, i.e., one for which a supervised learning problem can be built: for two of them (Covertypes, 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 1 summarizes the salient information of the four datasets.

**B. 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  $D$ , setting  $n = |\hat{D}| = |D|$ . Finally, we measured

the UP and DA indexes on the pair  $\hat{D}, D$ , using  $D_{\text{out}}$  as  $D_{\text{test}}$  when needed. 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,  $\beta$ -VAE, IntroVAE) five times for each partition of the single-table datasets, by varying the random seed for the random components of the approach (namely, the training of the VAEs). This way, we attempted to mitigate out possible artifacts in the results caused by lucky or unlucky conditions. We used the single proposed variant for the relational dataset partitions. Overall, we hence generated  $3 \cdot 3 \cdot 5 \cdot 5 + 1 \cdot 1 \cdot 5 \cdot 5 = 250$  synthetic datasets. In the tables presented below, we report the mean and standard deviation of the values of the indexes across the  $5 \cdot 5$  application of each variant to each dataset.

Concerning the UP 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.

Concerning the DA indexes, we proceeded as follows. For all the NNDR indexes (NNDR $_{\mu}$ , NNDR $_{\sigma}$ , and NNDR $_p$ ) and for NNDD, we proceeded as for CSE. For NNDD, we

**TABLE 2. Summary of the parameter values.**

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

performed the statistical significance test with two values of  $\alpha$ , 0.05 and 0.01.

We performed all the experiments by using the implementation of our pipeline described in Section IV-D 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 2 after some preliminary experiments.

### C. BASELINE

We remark that, to the best of our knowledge, no other methods exist that are capable of generating synthetic data in the same scenario our method is (heterogeneous data types, relational datasets). However, in order to provide a comparison baseline, we designed and implemented a simple methodology that applies to most of the cases of our scenario and experimented with it in the same way we did for VAE,  $\beta$ -VAE, and IntroVAE.

The baseline synthetic data generation technique works as follows. Given a single-table dataset  $D$  with a table  $T$ , this technique produces a compatible dataset  $\hat{D}$  consisting of a table  $\hat{T}$  that is just a subset of rows of  $T$ . Namely, while building  $\hat{T}$ , we select  $\frac{1}{2}|T|$  rows of  $T$  with random sampling without repetitions.

It can be seen that, in the trade-off between UP and DA, this technique favors UP. Since  $\hat{T}$  contains a subset of rows of  $T$ , it will be very likely useful for building an ML model whose performance is predictive of the one of an ML model built on  $T$ . On the other hand,  $\hat{T}$  clearly discloses the information in  $T$ , since it contains real data points. We hence expect this baseline to be challenging for our method in terms of UP. Concerning DA, regardless of the value of the corresponding indexes, this baseline technique does disclose real data points.

### D. RESULTS AND DISCUSSION

Tables 3 and 4 present the results for UP, respectively for the classification datasets, for which the target variable is categorical (Coverttype, 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 techniques and the effectiveness metrics used while evaluating MC are different for the classification (Accuracy and AUC) and regression cases ( $R^2$  and MSE). Table 5 presents the results for DA. All the tables show the

**TABLE 3. Results for UP on the classification datasets (mean  $\pm$  standard deviation across the five folds and five repetitions). For all the indexes, the lower, the better.**

D.	Method	MC-RF		MC-LRC		MC-ADA		MC-MLP		PCD	CSE
		Acc.	AUC	Acc.	AUC	Acc.	AUC	Acc.	AUC		
Cover.	VAE	0.01 $\pm$ 0.00	0.02 $\pm$ 0.00	0.01 $\pm$ 0.00	0.02 $\pm$ 0.00	0.01 $\pm$ 0.00	0.02 $\pm$ 0.00	0.02 $\pm$ 0.00	0.02 $\pm$ 0.00	0.01 $\pm$ 0.00	0.05 $\pm$ 0.01
	$\beta$ -VAE	0.01 $\pm$ 0.00	0.02 $\pm$ 0.00	0.01 $\pm$ 0.00	0.02 $\pm$ 0.00	0.01 $\pm$ 0.00	0.02 $\pm$ 0.00	0.02 $\pm$ 0.00	0.02 $\pm$ 0.00	0.01 $\pm$ 0.00	0.07 $\pm$ 0.02
	IntroVAE	0.30 $\pm$ 0.58	0.27 $\pm$ 0.57	0.28 $\pm$ 0.55	0.28 $\pm$ 0.52	0.29 $\pm$ 0.54	0.28 $\pm$ 0.54	0.49 $\pm$ 0.95	0.31 $\pm$ 0.59	0.01 $\pm$ 0.00	0.11 $\pm$ 0.01
	Baseline	0.00 $\pm$ 0.00	0.00 $\pm$ 0.00	0.00 $\pm$ 0.00	0.00 $\pm$ 0.00	0.01 $\pm$ 0.00	0.00 $\pm$ 0.00	0.00 $\pm$ 0.00	0.00 $\pm$ 0.00	0.00 $\pm$ 0.00	0.00 $\pm$ 0.00
Bank	VAE	0.12 $\pm$ 0.02	0.19 $\pm$ 0.15	0.23 $\pm$ 0.10	0.20 $\pm$ 0.15	0.10 $\pm$ 0.07	0.28 $\pm$ 0.08	0.07 $\pm$ 0.03	0.22 $\pm$ 0.14	0.00 $\pm$ 0.00	0.00 $\pm$ 0.00
	$\beta$ -VAE	0.03 $\pm$ 0.03	0.05 $\pm$ 0.00	0.05 $\pm$ 0.03	0.03 $\pm$ 0.01	0.03 $\pm$ 0.01	0.10 $\pm$ 0.02	0.03 $\pm$ 0.01	0.05 $\pm$ 0.00	0.01 $\pm$ 0.00	0.03 $\pm$ 0.00
	IntroVAE	0.01 $\pm$ 0.00	0.03 $\pm$ 0.05	0.01 $\pm$ 0.02	0.02 $\pm$ 0.04	0.02 $\pm$ 0.03	0.06 $\pm$ 0.09	0.02 $\pm$ 0.03	0.03 $\pm$ 0.06	0.01 $\pm$ 0.00	0.06 $\pm$ 0.01
	Baseline	0.00 $\pm$ 0.00	0.00 $\pm$ 0.00	0.00 $\pm$ 0.00	0.00 $\pm$ 0.00	0.01 $\pm$ 0.00	0.00 $\pm$ 0.00	0.01 $\pm$ 0.00	0.01 $\pm$ 0.00	0.01 $\pm$ 0.00	0.01 $\pm$ 0.00
Rel.	ML-VAE	0.04 $\pm$ 0	0.16 $\pm$ 0	0.08 $\pm$ 0	0.23 $\pm$ 0	0.17 $\pm$ 0	0.03 $\pm$ 0	0.03 $\pm$ 0	0.23 $\pm$ 0	0.08 $\pm$ 0	0.18 $\pm$ 0

**TABLE 4. Results for UP on the regression dataset (mean  $\pm$  standard deviation across the five folds and five repetitions). For all the indexes, the lower, the better.**

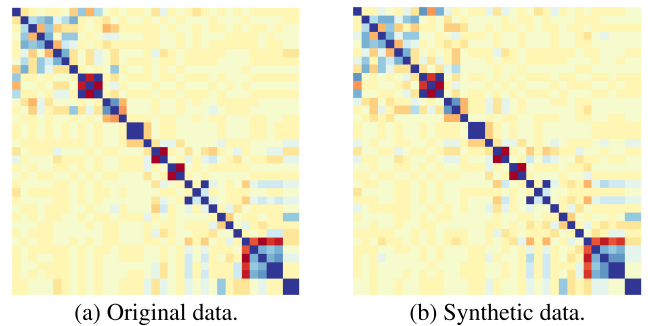
D.	Method	MC-LR		MC-RR		MC-SVR		MC-MLP		PCD	CSE
		$R^2$	MSE	$R^2$	MSE	$R^2$	MSE	$R^2$	MSE		
Ins.	VAE	0.06 $\pm$ 0.01	0.34 $\pm$ 0.01	0.06 $\pm$ 0.01	0.34 $\pm$ 0.01	0.08 $\pm$ 1.94	0.53 $\pm$ 0.03	0.18 $\pm$ 0.02	0.54 $\pm$ 0.02	0.01 $\pm$ 0.00	0.07 $\pm$ 0.01
	$\beta$ -VAE	0.03 $\pm$ 0.02	0.29 $\pm$ 0.05	0.03 $\pm$ 0.02	0.29 $\pm$ 0.04	0.24 $\pm$ 0.15	0.51 $\pm$ 0.12	0.15 $\pm$ 0.04	0.50 $\pm$ 0.07	0.01 $\pm$ 0.00	0.16 $\pm$ 0.02
	IntroVAE	0.03 $\pm$ 0.01	0.23 $\pm$ 0.03	0.02 $\pm$ 0.01	0.23 $\pm$ 0.03	0.17 $\pm$ 0.02	0.46 $\pm$ 0.04	0.12 $\pm$ 0.04	0.44 $\pm$ 0.07	0.02 $\pm$ 0.00	0.18 $\pm$ 0.01
	Baseline	0.00 $\pm$ 0.00	0.09 $\pm$ 0.00	0.01 $\pm$ 0.00	0.09 $\pm$ 0.00	0.00 $\pm$ 0.00	0.10 $\pm$ 0.00	0.02 $\pm$ 0.00	0.16 $\pm$ 0.00	0.01 $\pm$ 0.00	0.05 $\pm$ 0.02

variants of our method and the baseline applied to different datasets on rows and the values for the performance indexes on 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 and five repetitions (i.e., 25 samples) of each performance index. For all the indexes, with the exception of NNDD, the lower the better.

By looking at the tables, it can be seen that, in general, all UP values are very low, i.e., very good. In particular, for what concerns MC,  $\beta$ -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 Coverttype. For the 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 4 the correlation matrices for the original and synthetic data (after the numerical conversion, i.e., on  $X$  and  $\hat{X}$ ) for one fold-repetition of the Bank dataset with  $\beta$ -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

**FIGURE 4. Correlation matrices (of variables after the numerical conversion) of original and synthetic data for one fold-repetition of the Bank dataset with  $\beta$ -VAE. Red denotes negative correlation; blue denotes positive correlation; yellow denotes  $\approx 0$  correlation.**

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,  $\beta$ -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 Table 5 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

**TABLE 5. Results for DA on all the datasets (mean  $\pm$  standard deviation across the five folds and five repetitions). For NNDR indexes, the lower, the better; for NNDD indexes, the greater, the better.**

D.	Method	NNDR			NNDD	
		$\mu$	$\sigma$	$p$	0.05	0.01
Cover.	VAE	0.29 $\pm$ 0.02	1.25 $\pm$ 0.06	0.07 $\pm$ 0.01	0.20 $\pm$ 0.40	0.20 $\pm$ 0.40
	$\beta$ -VAE	0.30 $\pm$ 0.02	1.19 $\pm$ 0.06	0.02 $\pm$ 0.01	2.60 $\pm$ 0.80	4.20 $\pm$ 0.75
	IntroVAE	0.29 $\pm$ 0.01	1.04 $\pm$ 0.04	0.02 $\pm$ 0.01	1.20 $\pm$ 0.75	1.80 $\pm$ 1.47
	Baseline	1.00 $\pm$ 0.00	1.00 $\pm$ 0.00	1.00 $\pm$ 0.00	1.00 $\pm$ 0.00	1.00 $\pm$ 0.00
Bank	VAE	0.27 $\pm$ 0.02	1.23 $\pm$ 0.17	0.04 $\pm$ 0.00	0.00 $\pm$ 0.00	0.00 $\pm$ 0.00
	$\beta$ -VAE	0.28 $\pm$ 0.01	1.09 $\pm$ 0.11	0.04 $\pm$ 0.01	3.80 $\pm$ 0.75	4.60 $\pm$ 0.49
	IntroVAE	0.27 $\pm$ 0.01	0.99 $\pm$ 0.06	0.49 $\pm$ 0.00	2.60 $\pm$ 1.02	3.20 $\pm$ 0.98
	Baseline	1.00 $\pm$ 0.00	1.00 $\pm$ 0.00	1.00 $\pm$ 0.00	1.00 $\pm$ 0.00	1.00 $\pm$ 0.00
Ins.	VAE	1.07 $\pm$ 0.02	4.47 $\pm$ 0.18	0.02 $\pm$ 0.01	3.20 $\pm$ 0.75	4.40 $\pm$ 0.49
	$\beta$ -VAE	1.08 $\pm$ 0.05	4.25 $\pm$ 0.29	0.06 $\pm$ 0.00	0.20 $\pm$ 0.40	1.20 $\pm$ 0.40
	IntroVAE	1.04 $\pm$ 0.06	3.81 $\pm$ 0.38	0.07 $\pm$ 0.01	0.00 $\pm$ 0.00	0.00 $\pm$ 0.00
	Baseline	1.00 $\pm$ 0.00	1.00 $\pm$ 0.00	1.00 $\pm$ 0.00	1.00 $\pm$ 0.00	1.00 $\pm$ 0.00
Rel.	ML-VAE	0.10 $\pm$ 0	0.43 $\pm$ 0	0.02 $\pm$ 0	0.00 $\pm$ 0	0.00 $\pm$ 0

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 general very close to 0, regardless of the VAE variant, but the values for  $NNDR_\mu$  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 good (NNDD close to 1 for both values of  $\alpha$ ) for most of the variants and datasets.

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.

## VI. CONCLUDING REMARKS

In this paper, we presented a pipeline, based on VAEs, for generating synthetic tabular data from realistic business data with heterogeneous data types and missing values. We also introduced a set of indexes for measuring the effectiveness of our pipeline in generating synthetic data that is useful and prevent disclosure of information contained in the original data. We evaluated experimentally our method, using the aforementioned indexes, on three publicly available datasets consisting of one table each and on a custom relational dataset consisting on three tables. The results indicated that  $\beta$ -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.

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 work, 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.

## ACKNOWLEDGMENT

The experimental results in this article were obtained using publicly available data sets from the UCI Machine Learning Repository (see [1]).

The authors would like to thank the UCI Machine Learning Repository for making these data sets available.

## REFERENCES

- [1] D. Dua and C. Graff. (2017). *UCI Machine Learning Repository*. [Online]. Available: <http://archive.ics.uci.edu/ml>
- [2] The European Commission. (2019). *Shaping Europe's Digital Future*. [Online]. Available: [https://ec.europa.eu/info/strategy/priorities-2019-2024/europe-fit-digital-age/shaping-europe-digital-future\\_en](https://ec.europa.eu/info/strategy/priorities-2019-2024/europe-fit-digital-age/shaping-europe-digital-future_en)



- [3] (2018). *Towards a common European Data Space*. [Online]. Available: <https://eur-lex.europa.eu/legal-content/en/TXT/?uri=CELEX:52018DC0232>
- [4] (2016). *Opinion 03/2016 on the Evaluation and Review of the Eprivacy Directive (2002/58/EC)*. [Online]. Available: [https://ec.europa.eu/justice/article-29/documentation/opinion-recommendation/files/2016/wp240\\_en.pdf](https://ec.europa.eu/justice/article-29/documentation/opinion-recommendation/files/2016/wp240_en.pdf)
- [5] L. Rocher, J. M. Hendrickx, and Y.-A. de Montjoye, "Estimating the success of re-identifications in incomplete datasets using generative models," *Nature Commun.*, vol. 10, no. 1, pp. 1–9, Jul. 2019.
- [6] Y.-A. de Montjoye, L. Radaelli, V. K. Singh, and A. S. Pentland, "Unique in the shopping mall: On the reidentifiability of credit card metadata," *Science*, vol. 347, no. 6221, pp. 536–539, Jan. 2015.
- [7] D. 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 Now*, Tech. Rep., Jul. 2012. [Online]. Available: [https://papers.ssrn.com/sol3/papers.cfm?abstract\\_id=2076397](https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2076397)
- [8] A. Narayanan and V. Shmatikov, "How to break anonymity of the Netflix prize dataset," 2006, *arXiv:cs/0610105*.
- [9] N. Savage, "Synthetic data could be better than real data," *Nature*, Apr. 2023. [Online]. Available: <https://www.nature.com/articles/d41586-023-01445-8>
- [10] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, " $\beta$ -VAE: Learning basic visual concepts with a constrained variational framework," in *Proc. 5th Int. Conf. Learn. Represent. (ICLR)*, 2016, pp. 1–13.
- [11] H. Huang, "IntroVAE: Introspective variational autoencoders for photographic image synthesis," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018, pp. 1–12.
- [12] N. Park, M. Mohammadi, K. Gorde, S. Jajodia, H. Park, and Y. Kim, "Data synthesis based on generative adversarial networks," *Proc. VLDB Endowment*, vol. 11, no. 10, pp. 1071–1083, Jun. 2018.
- [13] H. Su, C. R. Qi, Y. Li, and L. J. Guibas, "Render for CNN: Viewpoint estimation in images using CNNs trained with rendered 3D model views," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 2686–2694.
- [14] S. Hinterstoisser, O. Pauly, H. Heibel, M. Martina, and M. Bokoeloh, "An annotation saved is an annotation earned: Using fully synthetic training for object detection," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshop (ICCVW)*, Oct. 2019, pp. 2787–2796.
- [15] S. Chatterjee, D. Hazra, Y.-C. Byum, and Y.-W. Kim, "Enhancement of image classification using transfer learning and GAN-based synthetic data augmentation," *Mathematics*, vol. 10, p. 1541, Apr. 2022.
- [16] M. Castelli, L. Manzoni, T. Espindola, A. Popovič, and A. De Lorenzo, "Generative adversarial networks for generating synthetic features for Wi-Fi signal quality," *PLoS ONE*, vol. 16, no. 11, Nov. 2021, Art. no. e0260308.
- [17] A. F. Oliveira, J. L. F. Da Silva, and M. G. Quiles, "Molecular property prediction and molecular design using a supervised grammar variational autoencoder," *J. Chem. Inf. Model.*, vol. 62, no. 4, pp. 817–828, Feb. 2022.
- [18] L. Xu, M. Skoularidou, A. Cuesta-Infante, and K. Veeramachaneni, "Modeling tabular data using conditional GAN," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 7335–7345.
- [19] L. Xu and K. Veeramachaneni, "Synthesizing tabular data using generative adversarial networks," 2018, *arXiv:1811.11264*.
- [20] L. Hu, J. Li, G. Lin, S. Peng, Z. Zhang, Y. Zhang, and C. Dong, "Defending against membership inference attacks with high utility by GAN," *IEEE Trans. Dependable Secure Comput.*, vol. 20, no. 3, pp. 2144–2157, May 2022.
- [21] M. Schulz and J. 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)," *Off. J. Eur. Union*, pp. 1–88, May 2016.
- [22] The European Commission. (2016). *General Data Protection Regulation (GDPR)*. [Online]. Available: <https://gdpr-info.eu/>
- [23] The European Commission, "Regulation of the European parliament and of the council: On European data governance (data governance act)," Eur. Commission, Tech. Rep., 2020.
- [24] K. Hänsch and L. A. 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," *Off. J. Eur. Union*, pp. 1–20, Nov. 1995.
- [25] 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)," *Off. J. Eur. Union*, pp. 1–88, May 2016.
- [26] 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*, vol. 19, no. 2, pp. 185–193, Jan. 2003.
- [27] D. Kingma and M. P. Welling, "Auto-encoding variational Bayes," in *Proc. 2nd Int. Conf. Learn. Represent. (ICLR)*, 2014, pp. 1–9.
- [28] L. Xu, M. Skoularidou, A. Cuesta-Infante, and K. Veeramachaneni, "Modeling tabular data using conditional GAN," 2019, *arXiv:1907.00503*.
- [29] D. Bouchacourt, R. Tomioka, and S. Nowozin, "Multi-level variational autoencoder: Learning disentangled representations from grouped observations," in *Proc. AAAI Conf. Artif. Intell.*, vol. 32, no. 1, pp. 1–8, Apr. 2018.



**DANIELE PANFILO** received the M.Sc. degree (cum laude) in industrial engineering from the Sapienza University of Rome, in 2014, the M.Sc. degree in knowledge engineering and data science from Maastricht University, in 2016, and the Ph.D. degree in deep-learning generative models from the University of Trieste, in 2022. He subsequently gained experience with applied ML in the healthcare and insurance industries as a Data Scientist with the Medtronic Bakken Research Centre, The Netherlands, and Allianz, Italy. He is currently the CEO and the Co-Founder of Aindo s.r.l., a startup dedicated to putting cutting-edge AI research into practice. His research interests include generative models, convolutional neural networks, image processing, and the applications of AI in healthcare and finance.



**ALEXANDER BOUDEWIJN** received the M.Sc. degree (cum laude) in knowledge engineering and data science from Maastricht University, The Netherlands, and the Ph.D. degree in engineering science from Katholieke Universiteit Leuven, Belgium, in 2023. At Aindo s.r.l., he conducts research and serves as a Science Communicator. His research interests include generative AI, data privacy, combinatorial optimization, complexity theory, and applications thereof in AI.



**SEBASTIANO SACCANI** received the M.Sc. degree (cum laude) in physics from the University of Trieste, and the Ph.D. degree from the International School for Advanced Study (SISSA), Trieste, in 2013, with a focus on the theory and simulation of condensed matter. He gained substantial working experience in the fields of mathematical finance and ML in the insurance sector. He is currently the CTO and the Co-Founder of Aindo s.r.l., where he specializes in generative models and AI consultancy across sectors.



**ANDREA COSER** received the M.Sc. degree (cum laude) in physics from the University of Padova, Italy, and the Ph.D. degree from the International School for Advanced Study (SISSA), Trieste, specializing in statistical physics. He subsequently a Postdoctoral Researcher with the Complutense University of Madrid, Spain, where his research topics included condensed matter physics and quantum computation. He has professional experience in AI and data science in application areas, such as health, safety, and environment. At Aindo s.r.l., he is essential with the Research and Development Department, at both the mathematical and the implementation level.



**BORUT SVARA** received extensive professional experience as a Freelancer in software engineering and data science. Subsequently, he joined Aindo s.r.l., as the Chief Technology Officer. He is also affiliated with the University of Trieste. His main responsibility is designing appropriate IT infrastructures for Aindo technologies. In doing so, he combines his background in software engineering with an advanced understanding of AI, ML, and database management. This facilitates a seam-

less and optimal integration of novel AI technologies into user-friendly software environments.



**CIRO ANTONIO MAMI** is currently pursuing the M.Sc. degree in data science and scientific computing with the University of Trieste. He has research experience in AI, ML, and optimization. At Aindo s.r.l., he conducts research into generative models, with a focus on domain-specific techniques for hyperparameter optimization. His research interests include image processing and high-performance computing.



**CARLO ROSSI CHAUVENET** received the M.Sc. degree (cum laude) in law and economics from Bocconi University, Italy, in 2004, the Ph.D. degree in civil law from the University of Padova, in 2008, the LLM degree in international trade and corporate law from the National University of Singapore, in 2009, and the LLM degree in corporate and financial services law from the New York University School of Law, in 2009. He is currently an Adjunct Professor in civil law with

Bocconi University. He is also a partner with CRCLEX, where he is a leading Expert in privacy and IT law. His research interests include data privacy and security, the law of internet technology, and IT law.



**ERIC MEDVET** (Member, IEEE) received the degree (cum laude) in electronic engineering and the Ph.D. degree in computer engineering from the University of Trieste, Italy, in 2004 and 2008, respectively. He is currently an Associate Professor in computer engineering, the Director of the Evolutionary Robotics and Artificial Life Laboratory, and the Co-Director of the Machine Learning Laboratory, University of Trieste. His research interests include evolutionary robotics and artificial life, evolutionary computation, and the applications of ML.

...