Tabular data generation using PINN technique

Jose Llanes Jurado

April 30, 2025

Abstract

The collection and use of data are becoming increasingly robust and widespread across various areas of daily life. However, there are contexts—particularly in the healthcare and psychological sectors—where the availability of patient or user data is limited. This scarcity constrains the applicability of statistics and machine learning (ML) methods due to small sample sizes, preventing the extrapolation of results and isolating potentially valuable studies.

In this context, Generative Adversarial Networks (GANs) have proven highly effective in generating realistic images, emulating different artistic styles or modifying existing visuals. Nevertheless, such training techniques can also be applied to other data formats, such as tabular data, in order to mitigate data scarcity in certain scenarios. The objective of this work is to study a method for generating tabular data capable of learning and reproducing the statistical distributions and correlations inherent in a real dataset. To achieve this, we employ the Physics-Informed Neural Networks (PINNs) approach, incorporating statistical information from the original dataset directly into the loss function of the deep learning (DL) model. Specifically, metrics such as the median and interquartile range are used for each variable, along with global dependency measures such as covariance and Pearson correlation, in order to preserve the structure and relationships of the original data. Results show potential in this approach, being the proposed models the once with higher performance comparing the model performance being trained with real data or generated and real. This new approach could be combined with different models of the state-of-the-art (SoA) to improve data generation.

1 Introduction

There are still very few studies specifically focused on the generation of synthetic tabular data, despite the growing need for large volumes of data to train increasingly complex machine learning (ML) models with a high number of parameters. It represents a significant obstacle, especially in contexts where the population sample is small, which limits the applicability of traditional statistical techniques and ML models, compromising the validity and generalization of the results obtained. Furthermore, factors such as legal restrictions, data protection policies, and the growing concern for user privacy make it difficult to share and reuse real data between studies, institutions, or companies. In this context, the generation of synthetic tabular data emerges as an essential tool, not only to expand existing data-bases but also to facilitate research, ensure the reproducibility of studies, and promote the development of innovative solutions without compromising the confidentiality of the original data.

Generative Adversarial Networks (GANs) are a class of deep learning (DL) training designed to generate new data samples that resemble a given dataset. Introduced by Goodfellow et al. in 2014 [1], GANs consist of two neural networks (NN), the generator and the discriminator, that compete in a zero-sum game. The generator aims to create realistic synthetic data, while the discriminator seeks to distinguish real data from generated samples, leading to a continuous improvement in the quality of the generated outputs. GANs have been widely used in various applications, including image synthesis, style transfer, data augmentation, and anomaly detection. They play a crucial role in fields such as computer vision, medical imaging, natural language processing, and even scientific simulations, where realistic data generation is essential. Due to their ability to learn complex data distributions and generate high-fidelity outputs, GANs have become a fundamental tool in modern artificial intelligence (AI) research and applications.

Several studies combine the generation of tabular data through GANs, such as the work by Ashrapov et al. (2020) [2] and the SDV library project [3]. Both approaches focus on data generation by initializing the input data randomly according to a certain distribution before passing

them to the generator model. The input data for the generator is randomly sampled according to the statistical distribution of the real data. In this way, the generative model only needs to modify the numerical values while preserving the input distribution in order to produce results that closely resemble the original dataset. However, the SDV library project does perform a post-processing step in which it analyzes the relationships between different columns through their covariance. However, this information is not transmitted at any point to the model or NN in order to allow it to learn the internal relationships between variables during the training.

Physics-Informed Neural Networks (PINNs) is a novel DL technique that integrates physical laws into the training process to enhance model accuracy and generalization. Unlike conventional data-driven approaches that rely solely on large labeled datasets, PINNs incorporate domain-specific knowledge by embedding physycal equations directly into the NN loss function. This approach enables PINNs to learn solutions that are consistent with underlying physical principles, reducing the need for extensive training data while improving interpretability and robustness. PINNs have been successfully applied across various scientific and engineering domains, including fluid dynamics, electromagnetics, and biomedical modeling, where they effectively solve forward and inverse problems with high accuracy. PINNs offer a flexible and scalable framework for solving complex, high-dimensional problems that traditional numerical methods struggle to handle efficiently.

The use of PINN is beneficial for a GAN because it could learn the hidden patterns of the data distribution of one and between different features. Metrics such as covariance matrix or correlation from the real data could be introduced directly to the GAN loss function as PINN technique does. Therefore, the model will learn patterns between the features as each relative variation or correlation. Other metrics or statistical values could be used as the median of each feature. The generator model will train trying to optimize those scores in the loss function and adjusting its generation to the real scores of the dataset.

This work aims to conduct an initial study on the incorporation of the PINN technique into the training of GANs, thereby combining both training paradigms in an effort to better capture not only the patterns of each variable individually but also the global relationships among them. To this end, the Materials and Methods section will define the datasets used and the loss functions that will be explored. The models proposed in this study will be compared against those from the SDV library. Lastly, all analytical methods used to evaluate the different models will be detailed. In the Results section, the best outcomes obtained for each dataset will be presented. Finally, the Discussion section will provide an in-depth analysis of the results and outline potential directions for future work.

All the code and development related with this research paper is implemented in the following github link TabularDataGAN.

2 Material and Methods

2.1 Dataset

Six datasets were used in this work, listed as follows. From the UCI repository, the datasets include: Adult, Student Dropout, Wine Quality, and Wisconsin Breast Cancer. From Kaggle, the datasets used are Maternal Health Risk and Titanic.

All of these datasets contain categorical columns. These categorical columns have been processed to be suitable for analysis. The processing followed is described below.

Given a categorical variable:

1. The frequency (ratio) of occurrence for each label in the categorical variable is computed.
2. Based on this ratio, intervals are created within the range [0, 1]. For example, for a categorical variable with three labels, the intervals would be $[0, x_1)$, $[x_1, x_2)$, and $[x_2, 1]$, where $x_1 < x_2$ and $0 < x_{1,2} < 1$.
3. For each interval, the midpoint is selected as $\mu = (x_i + x_j)/2$, and the standard deviation associated with the interval is defined as $\sigma = (x_i - x_j)/6$.
4. For each interval, a truncated normal distribution is generated which replaces the categorical labels of the variable.

Finally, Table 1 presents a descriptive analysis of the datasets used.

Table 1: Dataset summary

Dataset	Number of rows	Num. of columns	Percentage of Nans (%)	Num. categorical features	Unbalancement (%)
Adult	48842	16	0	8	23.93
Student dropout	4424	37	0	1	32.12
Wine quality	4898	12	0	1	96.26
Wisconsin breast cancer	569	32	0	1	37.26
Maternal Health Risk	1014	7	0	2	26.82
Titanic	891	12	8.1	6	40.62

Due to processing time constraints and the focus on studying reduced datasets, only the first 2000 rows will be used for datasets exceeding 2000 entries. Therefore, the Adult, Student dropout and Wine quality datasets will be limited to its first 2000 rows.

2.2 Cost function

This section describes, one by one, the various statistical functions used as part of the loss function to train the generative model. Throughout this work, the notation $\mathbf{X}$ will be used to represent the complete dataset, d will be used as a subscript to indicate a specific column, and D will denote the total number of columns. Appendix A contains more information about other cost functions implemented and tested but not used for the results presented in this work.

Interquantile Range (IQR)

The interquartile range is also considered a metric for the cost function, calculated according to equation 3

$$\mathbf{L}_{IQR} = \sum_{d=1}^{D} \left| \Big(Q(\mathbf{X}_{real,d}, 0.75) - Q(\mathbf{X}_{real,d}, 0.25) \Big) - \Big(Q(\mathbf{X}_{gen,d}, 0.75) - Q(\mathbf{X}_{gen,d}, 0.25) \Big) \right| \tag{1}$$

where $Q(x, \epsilon)$ denotes the extraction of the quantile of variable x at the threshold ϵ.

Covariance matrix

The covariance loss function is computed according with the covariance formula but only taking the upper diagional of the covariance matrix 2.

$$\mathbf{C}_{\text{upper}} = Triu\left(\frac{(\mathbf{X} - \bar{\mathbf{X}})^T (\mathbf{X} - \bar{\mathbf{X}})}{N-1} \right) \tag{2}$$

where $Triu()$ indicates to take the upper diagonal of the covariance matrix of the dataset $\mathbf{X}$. The covariance loss is computed making the difference between the upper diagonal of the real dataset and the generated dataset.

$$\mathbf{L}_{Cov} = \sum \left| \mathbf{C}_{\text{upper},real} - \mathbf{C}_{\text{upper},gen} \right| \tag{3}$$

Kernel density estimation (KDE)

The KDE function computes the KDE loss between two sets of features that could belong to distinct multivariate distributions. For each feature dimension, the probability density functions (PDFs) is estimated using Gaussian kernels through equation 4.

$$f_d(t; \mathbf{X}) = \frac{1}{Nh\sqrt{2\pi}} \sum_{i=1}^{N} \exp\left(-\frac{(t - \mathbf{X}_{i,d})^2}{2h^2} \right) \tag{4}$$

where N is the total number of samples and h is the kernel bandwith. Afterwards it is computed the absolute difference between two PDFs. The area under this difference curve (approximated using the trapezoidal rule) serves as a divergence measure between the distributions along each dimension. The final KDE loss is the sum of these divergence measures across all feature dimensions given by equation 5.

$$\mathcal{L}_{\text{KDE}} = \sum_{d=1}^{D} \int_{a_d}^{b_d} \Big| f_d(t; \mathbf{X}_{real}) - f_d(t; \mathbf{X}_{gen}) \Big| dt \tag{5}$$

where a_d and b_d are the integral bounds given by $a_d = \min(x_{i,d}, x_{j,d})$ and $b_d = \max(x_{i,d}, x_{j,d})$.

2.3 GAN-PINN

The cost function of the generative model is defined as shown in equation 6. In this formulation, the generator's loss is computed using a target array of ones, denoted as y_{ones}, which represents the label for real data. This is because the generator aims to produce data that is indistinguishable from real samples, thereby confusing the discriminator. The output of the discriminator when applied to the generated data, $\mathcal{D}(X_{gen})$, is compared against this target. The objective is to minimize the difference between the discriminator's prediction and the target label of one, effectively encouraging the discriminator to classify generated samples as real.

$$\mathcal{L}_G = f(y_{ones}, \mathcal{D}(\mathbf{X}_{gen})) \tag{6}$$

where normaly, $f()$ function is binary cross entropy (BCE) metric.

However, the main idea of this work is to compute the generator's loss by comparing the generated data against the real data. This comparison between the two datasets is carried out using a function $g()$, which essentially compares two matrices. In this way, similar to the approach used in PINNs (Physics-Informed Neural Networks), the cost function incorporates information about the target data that the model must replicate as faithfully as possible. The generated data is input into the loss function $\mathcal{L}_{G-PINN}$ together with the real data, and the loss is computed as follows:

$$\mathcal{L}_{G-PINN} = g(\mathbf{X}_{real}, \mathbf{X}_{gen}) \tag{7}$$

Additionally, a mixed cost function can be constructed, which is also analyzed in this work. In this case, the total loss is the sum of $\mathcal{L}D$ and $\mathcal{L}D - PINN$

$$\mathcal{L}_{Both} = l_D f(y_{ones}, \mathcal{D}(\mathbf{X}_{gen})) + l_{PINN} g(\mathbf{X}_{real}, \mathbf{X}_{gen}) \tag{8}$$

where l_D and l_{PINN} are coefficients that can be chosen by the experimenter. In this study, both were set to 1.

In the context of this study, the function $g()$ is defined as the sum of the IQR, the covariance matrix, and KDE. Various combinations of these metrics were explored in order to determine the configuration that best suited this work, with the *IQR-covmat-kde* combination proving to be the most effective. All the models studied in this work according with PINN technique will use this combination of metrics for the cost function.

2.4 Models

In this work, two DL models have been implemented. The first model is based on a fully connected NN with an encoder-decoder architecture (LinearNN). The model starts with 256 neurons, reduces to 128 neurons, and finally ends with 64 neurons in the latent dimension of the encoder-decoder. The decoder then follows a symmetric architecture, increasing to 128, 256, and finally matching the input data dimension. Each layer of this network consists of a linear NN with *BatchNormalization*, *ReLU*, and *Dropout*. The dropout rate is set to 0.1. This model will be tested using only PINN technique as a cost function, BCE as a cost function and both together in the loss, according to equations 6, 7 and 8. Therefore, the models will be denoted by *LinearNN PINN Loss*, *LinearNN BCELoss* and *LinearNN Both Loss* when the NN uses only PINN, BCE or both discriminators as a cost function.

The second model used is a one-dimensional convolutional neural network (CNN1D). This model also features an encoder-decoder architecture. It starts with an initial filter of 256, gradually decreases to 128 and 64, and ends with a latent dimension of 16 filters. The decoder then symmetrically increases

to 64, 128, and 256 filters. The encoder part is composed of CNN1D, while the decoder part consists of one-dimensional transposed convolutions (ConvT1D). Each layer of this network includes convolution operations, followed by *BatchNormalization*, *ReLU*, and *Dropout* with a rate of 0.1. All convolutions have a kernel size of 5. As *LinearNN*, the notation for this model differenciating by each loss function is *CNN1D PINN Loss*, *CNN1D BCELoss* and *CNN1D Both Loss* when the CNN1D uses only PINN, BCE or both discriminators as a cost function.

The discriminator model remains consistent throughout. This model starts with 512 neurons, followed by a cascade sequence of 256, 128, 64, and 16 neurons, ending with a single neuron for binary discrimination. Each layer contains a linear neural network, *BatchNormalization*, *LeakyReLU* with a negative slope of 0.2, and *Dropout* with a rate of 0.3.

2.5 SDV models experiment implementation

Additional models from the SDV library have been integrated into the analysis pipeline, allowing a comparison between the new models proposed in this work and the state-of-the-art models provided by the library. The models included are: *GaussianCopulaSynthesizer*, *CopulaGANSynthesizer*, *CTGANSynthesizer*, and *TVAESynthesizer*. All models were used with their default hyperparameters.

The implementation within the analysis pipeline involves training each model on the full dataset. Once a generator model has been trained, it is used in subsequent evaluation phases without being retrained or updated. To initialize each model, metadata describing the input data set must be provided, a step that is automatically handled by the SDV library.

2.6 Evaluation methods

Three different methodologies have been analyzed to evaluate the quality of tabular data generation.

2.6.1 Distinguish real from generated data through ML model

In this method, a sample of generated data is combined with a randomly selected sample of real data of the same length, forming a unified dataset. The goal is to predict which data points are generated and which are real. Therefore, a new column is added to the combined dataset, where the label for the generated subset is 0 and the label for the real subset is 1. A binary classifier is trained to evaluate its effectiveness in detecting real data. The objective of this method is to achieve the lowest possible results for the binary classifier, indicating that it is difficult to distinguish between real and generated data.

2.6.2 Train a ML using generated data

This method involves generating data and use it to fit a ML model in three different ways, the approaches are: training and testing on real data (Real Train), training the ML model with generated data and testing it on real data (Gen. Train), and training on a combination of generated and partial real data and testing on the remaining real data (Real-Gen. Train). The goal is to compare the metrics from the real training against the other two trainings, which use synthetic data. The objective is to achieve similar values in the metrics.

2.6.3 Statistical feature and dataset analysis

Statistical analysis is performed using non-normal tests to check if the distributions are similar to the original distribution. A threshold of 0.05 is used for the p-value; if the distributions have a p-value greater than 0.05, they do not show significant statistical differences and are considered the same distribution. Additionally, the *quality_report* metric from the SDV library is included.

2.6.4 Experimentation details

In all cases involving ML model studies, a logistic regression (LogR) model is used due to its simplicity and speed in fitting the data. For all methods studied, subsamples of real data are always used, with generated data being of the same length as the real data. This allows the studies to be repeated multiple times (since different subsamples are generated randomly). In all methods, a minimum of 50% of the original dataset length or a total of 1000 rows is used. Each methodology is repeated 30 times, and all metrics shown are averages over the 30 observations.

An automated pipeline has been constructed to evaluate all these metrics. (It might be useful to mention the duration and the equipment used for evaluation. This is important for two reasons: first, it ensures that all models and datasets are evaluated consistently; second, it allows for discussion on how preprocessing or more detailed studies could improve any of the models).

The complete results are shown in the results github page.

The whole analysis pipeline was runned in an AMD Ryzen 7 3700X 8-Core and it took in total 1.63 h.

3 Results

3.1 Distinguish real from generated data through ML model

The results in Table 2 show a single model per dataset—specifically, the best-performing model based on the highest accuracy score. Accuracy was chosen as the evaluation metric since the prediction target is consistently balanced at 50%. The table includes two models per dataset in cases where the best-performing model is not one of those developed in this study. This allows for a direct comparison between the metrics obtained by the SDV library models and those developed in this work.

The results indicate that the models from the SDV library achieve the lowest distinguishability between real and synthetic data. In particular, the GaussianCopulaSynthesizer consistently delivers the best results. The LinearNN PINN Loss model achieves the best performance for the Maternal Health Risk dataset. When comparing models across datasets, it can be observed that the performance differences between the best SDV model and the best model developed in this work are generally small. However, notable differences are observed for the Wine Quality and Student Dropout datasets. The average relative differences in accuracy, Cohen's kappa, and F1-score between the SDV models and those developed here are 10.09%, 41.93%, and 9.97%, respectively.

Table 2: Metric results for each model an dataset. The average and standard deviation are shown for each metric.

Dataset	**Model**	**Accuracy**	**Kappa**	**F1-Score**
Adult	*TVAESynthesizer*	0.6105 ± 0.0249	0.2218 ± 0.0498	0.5838 ± 0.0303
Adult	*LinearNN Both Loss*	0.6554 ± 0.0397	0.3111 ± 0.0792	0.6508 ± 0.0402
Maternal Health Risk	*LinearNN PINN Loss*	0.5648 ± 0.0418	0.1324 ± 0.0826	0.5599 ± 0.0451
Titanic	*GaussianCopulaSynthesizer*	0.5553 ± 0.0442	0.1157 ± 0.0864	0.558 ± 0.0492
Titanic	*LinearNN PINN Loss*	0.587 ± 0.0714	0.1774 ± 0.1406	0.5834 ± 0.0744
Wisconsin breast cancer	*GaussianCopulaSynthesizer*	0.5556 ± 0.0505	0.117 ± 0.0992	0.5628 ± 0.0635
Wisconsin breast cancer	*LinearNN PINN Loss*	0.5715 ± 0.0647	0.1509 ± 0.124	0.5678 ± 0.085
Wine quality	*GaussianCopulaSynthesizer*	0.5069 ± 0.0292	0.0181 ± 0.0574	0.5119 ± 0.0379
Wine quality	*LinearNN PINN Loss*	0.6843 ± 0.0261	0.3687 ± 0.052	0.6747 ± 0.0290
Student dropout	*TVAESynthesizer*	0.6770 ± 0.0231	0.3542 ± 0.0458	0.6512 ± 0.0284
Student dropout	*LinearNN PINN Loss*	0.7479 ± 0.0285	0.4957 ± 0.0564	0.725 ± 0.0336

3.2 Train a ML using generated data

Table 3 shows the results in terms of the data used for the training. The rows where the model used is empty is because the data used for the training is the real one. The model which generates the data that finally achieves the highest kappa score is shown in the table. In this case, kappa score is selected as the metric to order because the dataset could not be balanced an that could bias the selection of the best generator model.

As can be seen from Table 3 el model *LinearNN* es el modelo generativo que más aparece con un total de 8 veces, siendo 4 de ellas utilizando *Both Loss*, 3 veces *BCELoss* y 1 *PINN Loss*. En todos los casos se consiguen mejores resultados utilizando datos reales y sintéticos que solo sintéticos. En el caso de utilizar ambas fuentes de datos, pueden verse como los resultados en términos de accuracy varian desde un mínimo de -0.02% a 5.84% con respecto al modelo entrenado solo con datos reales. En el caso del dataset Winme quality, que es donde se alcanza el mínimo, se obtienen mejores resultados en accuracy y f1-score en este caso que solo entrenando con los datos reales, mientras que se empeora un 26.45% en el kappa score. En el caso de utilizar en el entrenamiento solo datos sintéticos estos resultados tienen mucha más variación, desde un mínimo de 1.42% a 15.25%

Table 3: Model results for the best model according with the type of training. Average and standard deviation is shown for each metric. The relative difference for each score against the training with the real dataset is shown inside parenthesis in terms of percentage (%).

Dataset	**Train step**	**Model**	**Accuracy**	**Kappa**	**F1-Score**
Adult	Real Train	-	0.8288 ± 0.0253	0.4645 ± 0.075	0.5652 ± 0.0681
Adult	Real-Gen. Train	*LinearNN Both Loss*	0.8108 ± 0.0270 (2.18)	0.4137 ± 0.0781 (10.94)	0.5245 ± 0.0712 (7.19)
Adult	Gen. Train	*TVAESynthesizer*	0.7480 ± 0.0333 (9.75)	0.3502 ± 0.0839 (24.60)	0.5177 ± 0.0693 (8.40)
Maternal Health Risk	Real Train	-	0.8635 ± 0.0349	0.6222 ± 0.0944	0.7090 ± 0.0781
Maternal Health Risk	Real-Gen. Train	*LinearNN Both Loss*	0.8544 ± 0.0376 (1.05)	0.6007 ± 0.1018 (3.46)	0.6942 ± 0.0852 (2.09)
Maternal Health Risk	Gen. Train	*LinearNN Both Loss*	0.8513 ± 0.0355 (1.42)	0.5960 ± 0.0983 (4.22)	0.6922 ± 0.0832 (2.37)
Titanic	Real Train	-	0.7978 ± 0.0518	0.5735 ± 0.1073	0.7402 ± 0.0708
Titanic	Real-Gen. Train	*LinearNN BCELoss*	0.7790 ± 0.0479 (2.35)	0.5328 ± 0.0990 (7.10)	0.7129 ± 0.0678 (3.68)
Titanic	Gen. Train	*TVAESynthesizer*	0.7678 ± 0.0470 (3.76)	0.5078 ± 0.0966 (11.45)	0.6980 ± 0.0677 (5.70)
Wisconsin breast cancer	Real Train	-	0.9484 ± 0.0287	0.8853 ± 0.0632	0.9257 ± 0.0425
Wisconsin breast cancer	Real-Gen. Train	*LinearNN PINN Loss*	0.9416 ± 0.0338 (0.71)	0.8698 ± 0.0754 (1.75)	0.9148 ± 0.0514 (1.17)
Wisconsin breast cancer	Gen. Train	*LinearNN Both Loss*	0.9276 ± 0.0364 (2.19)	0.8373 ± 0.0797 (5.43)	0.8919 ± 0.0550 (3.65)
Wine quality	Real Train	-	0.9556 ± 0.0141	0.1916 ± 0.1719	0.9771 ± 0.0074
Wine quality	Real-Gen. Train	*LinearNN BCELoss*	0.9558 ± 0.0157 (-0.02)	0.1409 ± 0.1629 (26.45)	0.9772 ± 0.0083 (-0.01)
Wine quality	Gen. Train	*LinearNN BCELoss*	0.8851 ± 0.1206 (7.38)	0.1244 ± 0.1392 (35.07)	0.9319 ± 0.0896 (4.63)
Student dropout	Real Train	-	0.7440 ± 0.0315	0.2795 ± 0.0794	0.4197 ± 0.0772
Student dropout	Real-Gen. Train	*TVAESynthesizer*	0.7005 ± 0.0357 (5.84)	0.2939 ± 0.0747 (-5.16)	0.5084 ± 0.0590 (-21.14)
Student dropout	Gen. Train	*TVAESynthesizer*	0.6305 ± 0.0384 (15.25)	0.2458 ± 0.0699 (12.04)	0.5220 ± 0.0527 (-24.37)

3.3 Statistical feature and dataset analysis

Table 4 presents the statistical and descriptive results of the various models for each dataset. In this case, the model achieving the highest QR score from the SDV library is considered the best model for that dataset. The table reports the mean QR score along with the percentage of variables for which more than 50% of the generated columns are not statistically different from the corresponding real columns in the same dataset.

In this analysis, the best-performing models tend to be those from the SDV library, particularly

the TVAESynthesizer. The QR score is consistently above 0.85, with the highest score reaching 0.9547 in Wine Quality dataset. Moreover, the percentage of variables with no statistically significant differences in the generated data is always greater than 0%, and reaches as high as 100% for the Wine Quality dataset. On the other hand, the PINN approach is also present in all the models proposed in this work. Indeed, the best model obtained for dataset Wisconsin breast cancer is the proposed *LinearNN Both Loss*. The rest of the results presented are close to the library SDV for the case of Maternal Health Risk, Titanic. However very different results are obtained for dataset Adult, Wine quality and specially Student dropout, where the QR score achieved is much lower.

Table 4: Statistical model results for the best model for each dataset according with QR score.

Dataset	**Model**	**Mean SDV QR score**	**Features above .05 p-value (%)**
Adult	*TVAESynthesizer*	0.9049 ± 0.0026	57.14
Adult	*LinearNN PINN Loss*	0.8164 ± 0.012	14.29
Maternal Health Risk	*TVAESynthesizer*	0.8866 ± 0.0045	71.43
Maternal Health Risk	*LinearNN PINN Loss*	0.8846 ± 0.0114	71.43
Titanic	*GaussianCopulaSynthesizer*	0.8974 ± 0.0056	88.89
Titanic	*LinearNN Both Loss*	0.8918 ± 0.0096	66.67
Wisconsin breast cancer	*LinearNN Both Loss*	0.9006 ± 0.0154	83.33
Wine quality	*GaussianCopulaSynthesizer*	0.9546 ± 0.0036	100
Wine quality	*LinearNN Both Loss*	0.9015 ± 0.0232	25
Student dropout	*CTGANSynthesizer*	0.8609 ± 0.0051	20
Student dropout	*LinearNN Both Loss*	0.4551 ± 0.0074	10

4 Discussion

The results of this study show that the models developed, along with the proposed technique, do not always achieve the best performance. However, they tend to be the most effective methodology for training machine learning models, being the model type that appears most frequently across datasets. Even though the proposed approach is only once the best at making real and synthetic data indistinguishable, the data generated using the PINN technique consistently shows results very similar to those produced by SDV models. Furthermore, this study provides a fully automated and general evaluation of all models, although a more exhaustive and tailored analysis is necessary to determine the optimal model for generating tabular data.

The findings highlight a new training paradigm for GANs that can complement the canonical training approach of such generative techniques. Although the results do not show a consistent improvement across all tests, they are comparable to those achieved by well-established and extensively validated state-of-the-art methods. Moreover, the PINN-based technique proposed in this study can be integrated into any generative model. The cost functions introduced here—or alternative ones—could be implemented in other types of neural networks to enhance robustness and provide additional

information about the dataset to be generated.

The model with best results of the proposed models is *LinearNN Both Loss* which has overcome in some cases the puntuation of SDV library. However, there is also space for improvement and further exploration of the cost function metrics to enhance the learning process of the generative model. In this work, three cost functions were used: $\mathcal{L}_{IQR}$, $\mathcal{L}_{Cov}$, and $\mathcal{L}_{KDE}$. Other types of cost functions could be implemented either across various datasets or tailored to a specific one, allowing the development of more customized loss functions suited to each case study. Additionally, training could benefit significantly from the implementation of normalized cost functions. For example, if both the discriminator and PINN components of $\mathcal{L}_{Both}$ were constrained within a range (e.g., 0 to 1), the coefficients l_D and l_{PINN} would then be more interpretable and meaningful within the overall cost function.

The use of the PINN technique is not only beneficial for tabular data, as demonstrated in this work, but it can also be applied to the construction of GANs for generating text or images. Introducing known patterns from the dataset or the data distribution can facilitate the training process and result in more realistic data generation. For instance, in text generation, certain linguistic rules known a priori, such as the tendency for a vowel to follow a consonant or the average word length, can be incorporated. In the case of image generation, the approach can be tailored to the specific type of images being generated. Metrics could be introduced to ensure smooth color gradients, avoiding abrupt transitions between colors, which would be a realistic scenario.

5 Conclusion

This work represents a first step in the application of the PINN technique for tabular data generation. The use of this approach has proven to be beneficial for generating tabular data aimed at robust training of machine learning models. This type of technique can also be integrated and combined with other generative models, including those provided by the SDV library itself. Furthermore, it has the potential to be extended to other domains such as image or language generation, by designing appropriate loss functions tailored to those tasks.

References

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

[2] Insaf Ashrapov. Tabular gans for uneven distribution, 2020.

[3] Neha Patki, Roy Wedge, and Kalyan Veeramachaneni. The synthetic data vault. In *IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 399–410, Oct 2016.

Appendix A

Other Cost functions

Other cost functions used are median, Pearson correlation and autocorrelation.

Median

The first cost function used is the difference, for each variable, between the median of the real and generated dataset. This comparison is defined by Equation 9.

$$\mathbf{L}_{Med} = \sum_{d=1}^{D} \left| Med(\mathbf{X}_{real,d}) - Med(\mathbf{X}_{gen,d}) \right| \tag{9}$$

where $Med()$ denotes the median operation.

Pearson correlation

The Pearson correlation of a matrix is computed as follows:

$$\mathbf{R}_{\text{upper}} = Triu\left(\frac{\mathbf{C}}{\sigma_i \sigma_j}\right) \tag{10}$$

where $\mathbf{C}$ is the covariance matrix and σ_x is the standard deviation of the feature x. And the loss function of the correlation matrix is computed as equation 11 shows

$$\mathbf{L}_{Corr} = \sum \left| \mathbf{R}_{\text{upper},real} - \mathbf{R}_{\text{upper},gen} \right| \tag{11}$$

Autocorrelation

Autocorrelation is computed for each feature as equation 12.

$$ACF = \frac{\sum_{i=1}^{N}(x_i - \bar{x})(x_{i-1} - \bar{x})}{\sum_{i=1}^{N}(x_i - \bar{x})^2} \tag{12}$$

Where the loss function is specify in equation 13.

$$\mathbf{L}_{ACF} = \sum_{d=1}^{D} \left| ACF_{d,real} - ACF_{d,gen} \right| \tag{13}$$