Received 18 October 2023, accepted 3 January 2024, date of publication 15 January 2024,
date of current version 25 January 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3354277

RESEARCH ARTICLE

A Methodology and an Empirical Analysis to Determine the Most Suitable Synthetic Data Generator

A. KIRAN AND S. SARAVANA KUMAR, (Member, IEEE)
Department of Computer Science and Engineering (CSE), School of Engineering and Technology (SOET), CMR University, Bengaluru, Karnataka 562149, India
Corresponding author: A. Kiran (kiranarnady@gmail.com)

ABSTRACT According to a report published by Gartner in 2021, a significant portion of Machine Learning (ML) training data will be artificially generated. This development has led to the emergence of various synthetic data generators (SDGs), particularly those based on Generative Adversarial Networks (GAN). All research endeavors so far have been exploratory, focused on specific objectives such as validating utility or disclosure control or assessing how generators can decrease or increase inherent bias with differential privacy. Hence, we aim to empirically identify an AI-based, data generator that can produce datasets that closely resemble real datasets, while also determining the hyper-parameters that enable a satisfactory balance between utility, privacy, and fairness in the datasets. To achieve this, we utilize the Synthetic Data Vault, Data Synthesizer, and Smartnoise-synth, which are three synthetic data generation packages that are accessible via Python. Different data generation models available within the package are presented with 13 tabular datasets iteratively as sample inputs to generate synthetic data. We generated synthetic data using every dataset and generator and investigated the goodness of the generator using five hypothetical scenarios. The utility and privacy offered by the generated data were compared with those of real data. The fairness in the ML model trained with synthetic data was used as a third metric for evaluation. Finally, we employ synthetic data to train regression and classification Machine Learning (ML) algorithms and evaluate their performance. After conducting experiments, analyzing metrics, and comparing ML scores across all 11 generators, we determined that the CTGAN from SDV and PATECTGAN from the SN-synth package were the most effective in mimicking real data for all 13 datasets utilized in our research.

INDEX TERMS Synthetic data, synthetic data vault, data synthesizer, SmartNoise-synth, GAN, VAE.

I. INTRODUCTION

The combined utilization of Machine Learning (ML) and Artificial Intelligence (AI), known as AI-ML, has become increasingly prevalent in recent times to gain insights from data, predict outcomes, analyze trends, make decisions, and provide potential solutions to problems. These models were trained using data, with the patterns within the presented data serving as the basis for prediction. Deep learning models, on the other hand, can adapt and learn independently from data. The reliability and effectiveness of AI-ML models depend on the quality and accessibility of the data utilized during the training process. Challenges faced by ML models include restrictions on data privacy, high costs associated with data collection, the quality of the data utilized, and potential biases inherent within the data. In 1993 Donald B Rubin [1] introduced synthetic data. An early study comparing synthetic data with real data [2] has showed that the statistical inferences from synthetic data and the original data match. This appears to overcome the challenges of data availability. The concept of Differential Privacy (DP) [3] was introduced into synthetic data generation [4] to improve disclosure control. Recent works [5], [6], [7] have focused on the utility and reliability of machine learning algorithms trained on synthetic data and have also produced encouraging results, addressing challenges such as privacy protection,

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

data collection costs, and data quality. According to Gartner, synthetic data will constitute 70% of the data used in AIML training by 2030, leading to the emergence of several commercial synthetic data generators (SDG) such as Gretel.ai, Datagen, MostlyAI, and CVEDIA. Despite progress in synthetic data, the persistent problem that challenges AI-ML model training is the inherent bias in the data. In the real world, bias can occur and be attributed to factors such as gender, age, demographics, race, and even the physical attributes of objects such as their shape, colour, or size. However, attempting to address this issue through techniques such as oversampling or under-sampling of data using SMOTE [8] or its variants may not necessarily result in a useful ML model as they tend to focus solely on the target class and disregard subgroups within each feature. Furthermore, the implementation of DP to protect the privacy of datasets may lead to a disparate impact [9], [10], [11], [12] on the resulting dataset, particularly in imbalanced cases.

The reliability of synthetic data remains unclear, leading to questions on the most effective option for generic use across all tabular data samples. We take three key metrics into consideration for any synthetic dataset deemed suitable for training machine learning algorithms. It is collectively termed as evaluation metrics in this document.

1) **Utility** [13]**:**The ability of the dataset to maintain the same statistical structure as that of the original sample from which it was generated.
2) **Disclosure control or privacy** [14]**:**How much privacy protection the synthetic dataset can offer without disclosing the identity or confidentiality of any individuals in the dataset? (E.g.: Name, City, Date of Birth, Phone Number etc)
3) **Fairness** [15]**:**Equal treatment of all groups, subgroups, class, individuals in the dataset

Achieving an ideal balance between data utility, privacy, and reduced bias is a significant challenge. The parameters that generate acceptable results in generative models are yet to be fully understood. Existing literature focuses on various aspects such as the preservation of data utility, the role of differential privacy in achieving disclosure control, and the impact of differential privacy on the balance of data that results in the widening gap between majority and minority groups. Despite these efforts, there is no conclusive evidence to support the superiority of any SDGs in achieving the right balance of evaluation metrics, including utility, privacy, and fairness in machine learning. Our experiments are directed towards the identification of an SDG and its hyper-parameters that produce synthetic data with a resemblance to real data, regardless of the input sample's characteristics. This generator must yield a satisfactory score while being utilized to train the classification and regression machine learning algorithms and maintain an appropriate balance of evaluation metrics. The objective was achieved by following a methodical approach to answering the following questions.

1. We begin by determining how the data generators treat the majority and minority target classes in input data samples that have different dimensionality and datatypes (continuous, categorical, constants, and discrete)
2. Next, we checked whether increasing the number of samples during data generation affects the quality of the generated dataset.
3. Does pre-processing (oversampling the minority class) of the data before being fed to the generator have any impact?
4. If the SDGs produce the datasets for all 13 datasets without missing any categorical values, we then evaluate which synthetic data generator produce a dataset that matches or closely matches the real dataset in terms of its utility, disclosure control, and fairness.
5. Finally, we determine the optimum parameter tuning that must be used during dataset generation.

The subsequent sections of this paper are arranged as follows. Section II discusses previous research on the verification and comparison of factual data using different techniques. Section III encompasses the approach taken and the experimental configuration, which also includes the sample datasets utilized. Section IV presents the outcomes of the study and finally, the discussion and conclusions are presented in Section V.

II. RELATED WORK

The utility of synthetic datasets in machine learning has been thoroughly examined using various data generators, as evidenced in previous studies [7], [16], [17]. The effectiveness of synthetic data in machine learning has also been validated [5], [6], [18]. Additionally since the inception of Generative Adversarial Networks [19] by Ian Goodfellow, several innovators have devised data generation techniques using GAN [20], [21], [22], [23].

A Generative Adversarial Network (GAN) comprises two models: generative and discriminative. The former endeavors to replicate the original data by introducing noise, whereas the latter compares the generated data against the original data to determine the degree of similarity in terms of their distribution. A shortcoming of GAN is that it considers the class label as an additional attribute, which leads to difficulties in the classification model. However, the Conditional GAN (CGAN) [21] resolves this issue by handling the class labels separately, thereby improving the quality of data for the classification model. Owing to the deep networks utilized in GAN, the system attempts to recall the training data and improve it to achieve the closest resemblance to the original data. This results in the possibility of revealing sensitive or personal information that may be present in the original data. To address this challenge, DPGAN [24] incorporated differential privacy (DP) into GAN. The approach used by the DPGAN is to train the discriminator with noise-induced data and have the generator predict based on it, leading to differentially private synthetic data. The PATE-GAN [20] model takes this idea further by applying the Private Aggregation of Teacher Ensembles (PATE) to the GAN. PATE-GAN employs a teacher and student model to produce a noisy dataset that is then used to train the discriminator, resulting in more effective disclosure control than DPGAN.

It is well-established that Differential Privacy (DP) is an effective tool for safeguarding the privacy of released datasets. However, it also affects the utility and fairness of synthetic datasets. Therefore, it is crucial to analyze the effect of DP on synthetic data, particularly when disparities exist between majority and minority classes in the dataset. Various versions of Generative Adversarial Networks (GANs) have been employed to demonstrate the unequal impact of DP on balancing classes in synthetic datasets. Ganev et al. utilized three different datasets and employed PrivBayes, W-GAN, and PATE-GAN to generate synthetic data to showcase the differences. They deduced from their findings that PrivBayes decreases the gap between the minority and majority classes, whereas PATE-GAN increases it. However, the results obtained from W-GAN were inconsistent. The results and observations were similar when DP was applied to synthetic data generation for healthcare datasets using GAN [15]. Karan Bhanot et al, delved deeper into developing a robust metric for measuring the fairness of synthetic healthcare data. They also asserted the necessity of introducing fairness in the dataset during data generation. The reason for the impact on fairness has been attributed to the DP Stochastic Gradient Descent, gradient clipping, and additive noise that introduce bias in the dataset resulting in unfavorable outcomes of the machine learning model trained on these datasets [11], [12]. Even if the training data set has a small disparity among its classes, DP can result in a larger imbalance in the resulting synthetic data set [25]. Pre-processing of the dataset by performing multi-label under-sampling until both the majority and minority numbers are equal provides better fairness in the generated data [26]. Blake Bullwinkel et al demonstrated the effect of pre-processing using four different SDGs. They used the Multiple Weights Exponential Mechanism (MWEM), DP-CTGAN, PATECTGAN, part of the Smartnoise-synth (SN-synth) package, and MST as data generators. The observation made in this study is that GAN-based generators produce varying results. Blake Bullwinkel et al suggested hyperparameter tuning. The efforts to benchmark [23] four different differentially private GAN-based SDGs showed that PATE-GAN offered better results when the privacy budget € was $\geq$ 3.0 and DPCTGAN offered better results when € was $\leq$ 1.0. The conclusion does not generalize to which of the generators is better. Bias mitigation options using the privatized likelihood ratio also highlight that bias-related challenges remain in the SDGs [27].

Our work differentiates the approach of examining SDGs against the three metrics across a range of datasets. This involved conducting comparisons, analysis, and identification of the generator, using 13 tabular datasets with varying dimensions and characteristics. Our methodology includes the use of GAN-based generators with hyperparameter tuning, and varying privacy budgets (€). Data synthesizer was used to compare disclosure control. Additionally, we employed oversampling of the original dataset and a combination of generators to generate data, followed by an evaluation of their respective metrics.

III. METHODOLOGY

Fig 1 illustrates the approach employed in our experiments, which incorporates the design of experiments and replication principles to produce numerous synthetic datasets from the initial data samples. Thirteen tabular datasets with diverse attributes were utilized. The dataset with the lowest number of rows was malware dataset which contained 374 rows. On the other hand, credit card datasets had the largest number, totaling 284808. Out of the thirteen datasets utilized, eight of them had several records ranging from 1000 and 62000. To avoid the extremes and ensure a fair comparison, a maximum sampling size of 50,000 was established. This sampling size was then employed to assess performance using the learning curve. The objective was to ascertain whether the sample size had any impact on the resulting synthetic datasets. Upon examining the learning curve provided in Fig 11, it becomes apparent that the accuracy of most datasets levels off after reaching 20,000 records, except for the gaussian and TVAE datasets. Hence, the training sample size was set to 2000 on the lower end in the initial iteration which was 4% of the maximum size and then increased to 5000 (10%), 10000 (20%), 20000 (40%) and 50000 (100%) for every dataset.

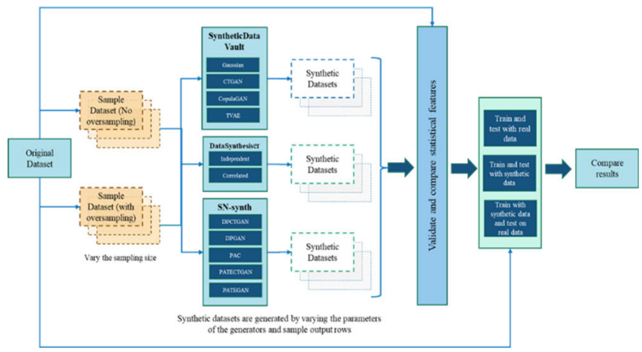


FIGURE 1. The design of experiment and replication methodology adopted for experiments to generate and evaluate synthetic data generated using different generators.

To generate, compare, and assess the SDGs, 11 different synthetic data generators available as a part of Python packages were utilized. These include the Synthetic Data Vault [28] (SDV), DataSynthesizer [4] (DS), and SmartNoise-Synth as well as one commercial generator (GretelAi). For SDV, the parameters EPOCH and BATCH_SIZE are altered, whereas for DS, the privacy attribute € is varied to introduce noise. For the SmartNoise-Synth generators, the privacy budget € is adjusted. The data generators that produce datasets that are sufficiently close to the real data are identified, and subsequently, the hyperparameter is modified for the most effective model to obtain the optimal parameters.

The generated synthetic data are then employed to train a variety of classification and regression ML algorithms including XGBoost, Support Vector Machine, Logistic Regression, KNeighbors, and Random Forest. It is difficult to choose an ML algorithm that fits all types of data. Therefore, five distinct classification and regression algorithms were chosen. XGBoost algorithm is well suited for datasets of significant size, while SVM is appropriate for smaller datasets. Logistic Classifier and Regression work when the data exhibits linearity. KNN operates based on the similarity and it is used to achieve high accuracy. Additionally, the algorithms introduce varying degrees of bias. Consequently, to ensure that our observations and conclusions are not solely reliant on a single algorithm, we employed the most optimal classification and regression algorithms in our experiments. The dataset was split into 70% training data and 30% testing data. Initially, the machine learning algorithm was trained using the original data and a baseline was established for comparison. In the second phase, the synthetic data were split with a ratio of 70:30. Finally, in the third phase, the algorithm was trained using 70% of synthetic data and tested against the original data, which served as an indicator of the actual performance of the synthetic data. An assessment of the generator comparison was executed via the collection of various parameters for the classification and regression algorithms. A generator comparison was performed by capturing different parameters for the classification and regression models.

Apart from comparing the scores of the machine learning models the evaluation metrics of the synthetic data were compared using following measurements. The utility of synthetic data was measured using visual as well as quantitative measures. Visual comparison was through density distribution (confidence interval overlap) [13] and correlation graphs for classification data and regression plots were used for regression data. Quantitative measurements included KL Divergence, Euclidean distance, and Hellinger distance to compare the similarities of real data and synthetic data.

Kullback-Leilber Divergence: Also known as relative entropy, it measures the disorder (entropy) in the sample being compared. It can be applied for both numerical as well as categorical values.

Euclidean Distance: It simply measures the point-to-point distance between two data points being compared.

Hellinger Distance:measures the difference between two probability distributions.

It is used as a common metric for both classification and regression models. In addition, scatter plots were used for visual comparison of the regression models. To verify privacy protection, a technique known as the "concept of uniqueness" [29] was used. The process of identifying unique data within a dataset involves utilization of a combination of attributes. The likelihood of an individual successfully identifying a specific record based on known parameters can be measured using this method. The specific utility [17] of the dataset was then measured using the confusion matrix and ROC score. The Model score, Mean Squared Error (MSE), Mean Aggregate Error (MAE), Root Mean Squared Error (RMSE) and R Squared (R2) are used as utility metrics for the regression model. The ratio of majority to minority classes in the dataset is used to determine the effectiveness of the SDG in generating the target class which is the minority. Finally, the bias introduced by the ML classifier algorithms as a result of training with synthetic data and bias introduced by real datasets were compared using dalex [30], an package in Python to check fairness.

IV. EXPERIEMENTAL SETUP

A. DATASET

Table 1 lists the datasets utilized as exemplars for the SDGs. These data were meticulously selected based on following criteria upon which we wanted to understand and compare the synthetic data generators. Eleven of the datasets were datasets with categorical class variables whereas two of them were continuous target variables. Criteria for selection were,

1. The target class distribution and imbalance in them. This was to evaluate how the SDGs would treat the minorities in the dataset. (e.g., stroke, credit card approval, credit card fraud)
2. Number of categorical columns that represented constant values. (e.g., wafer anomaly and malware dataset)
3. Correlation among attributes of the data
4. Dimensionality (number of columns) varying from very low to very high. (e.g., Diabetes 9 columns, wafer anomaly dataset with 1559 columns)
5. Variation in the number of records in the dataset. (e.g., malware 324 records and credit card fraud with 284808 records)
6. Datasets having some attributes through which we can demonstrate the data privacy (e.g. Adult income, HRA, Cerebral stroke)

The first three attributes of a dataset would be a challenge for any machine learning. Therefore, the dataset was intentionally picked to evaluate how the SDGs would perform with such challenging datasets. Attributes 5 and 6 were to understand the impact of the size of the dataset on privacy and performance of the generators.

The number of datasets chosen was intentionally high to facilitate comparative analysis and identification of the optimal generator with considerable generalizability. The synthetic data generated were subjected to training the ML classifier algorithm and later validated with a regression algorithm to ensure that the results obtained, and observations made for a specific generator did not vary much. All the datasets used were obtained from the public website www.kaggle.com

B. SETUP

The experiments were conducted using Python version 3.8.13, by establishing an Anaconda environment with a conda version 22.9.0. To generate synthetic data, freely available Python packages Synthetic Data Vault (SDV) 0.17.1, Datasynthesizer (DS) 0.1.11, and Smartnoise-synth (SN-synth) 0.3.5.1 were utilized. Additionally, Gretel-synthetics 0.18.1, a commercially available Synthetic Data.

Generator (SDG) package was used to generate data for a few datasets and compare its performance with the freely available packages in Python. Dalex 1.5.0 facilitated the assessment of the fairness of the classifiers. Synthetic data were generated on the Azure ML studio, utilizing Windows virtual machines with 8-core and 4-core CPUs, with 16 GB RAM and 8 GB RAM, respectively. All validations were executed using an Intel 2.4GHz dual-core processor with 12 GB RAM.

C. EXPERIEMENTS

SDV offers various models, namely Gaussian, CTGAN, CopulaGAN, and TVAE, to generate data. When using Data Synthesizer, the options available for generation include Random, Independent and Correlated options. For our experiments, we used Independent and Correlated options. SN-synth employs generators such as DPCTGAN, DPGAN, PAC, PATECTGAN, and PATEGAN. Fig 2 illustrates the workflow for the process, which involved using different sample sizes as input. Each iteration consisted of configuring the key parameter for each generator, which determined the dataset's quality. Our objective was to start with a smaller value and then incrementally increase it while continuously evaluating the dataset. We terminated the iteration process once we achieved an acceptable ROC score for classification or when the scores began to deteriorate. In the case of SDV, the identified parameters were epoch and batch_size. In the first iteration, the epoch value was set to the minimum of 10 and then incrementally increased to 50, 100, 200, 250, and 300. Similarly, the batch_size was set to 50, 100, 200, 300, and 500. Depending on each iteration's performance, the epoch values randomly varied between 10 and 300, ultimately stopping at 300 as moving beyond 300 did not show much improvement in the metrics. Anonymization was not employed for the SDV generators. For the Privacy attribute in DS, the variation is performed incrementally [€: 0.1, 0.5, 1.0, 5.0, and 10.0] for both the Independent and Correlated generation. Concerning SN-synth-based generators, only the privacy budget € is varied between [1, 5, 10, 20, and 50] in the first pass. The default values of 2e-04 for the generator learning rate, 1e-06 for discriminator decay, epoch set to 300, and batch_size set at 500 are maintained. The parameters were varied in the subsequent iterations.

To find answers to other questions, a few additional variations were attempted. First of these, shown in Fig 3, oversamples the minority target class using SMOTE with parameter, *sampling_strategy* set to *"auto"*. Oversampled data were used as input samples for the generators to generate synthetic data. The rest of the process remains the same as shown in Fig 2.

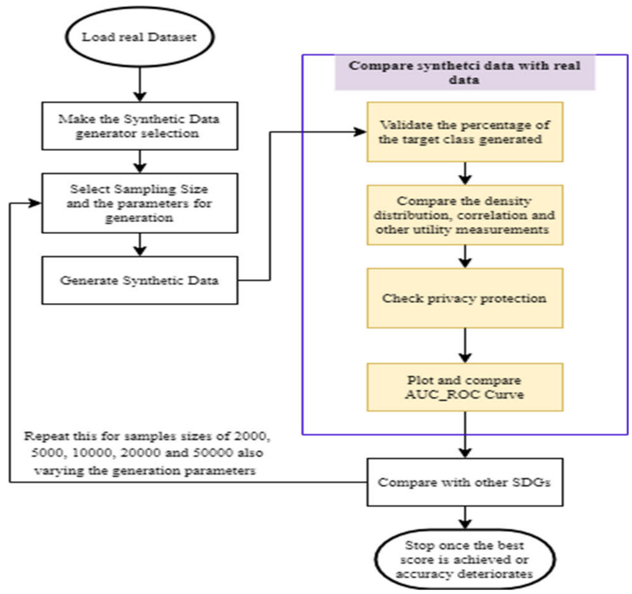


FIGURE 2. **The iterative process followed for generating synthetic data by changing the sample sizes and parameters for the data generation.**

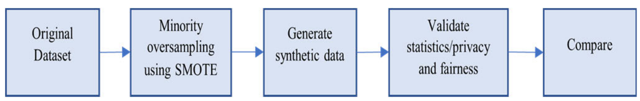


FIGURE 3. **Synthetic data generation by oversampling the minority class in the input sample.**

The DPCATGAN and PATECTGAN have additional parameters that influence the quality of the generated synthetic data. Therefore, the SDGs were further tuned by varying their hyperparameters. In the case of DPCTGAN, the gradient of noise was determined by the parameter Sigma, which was set to a default value of 5 in the initial pass. in the initial pass. Subsequently the values were changed to 1, 2, 3, and 10 and the results were evaluated. The effect of these variations is discussed in the next section. In contrast, PATECTGAN utilizes *noise_multiplier*, *student* and *teacher* iterators as parameters. In addition, *generator_lr* and *discriminator_lr* were used. To achieve moderate data privacy, all tests were performed using privacy budget of € =5. The other parameters were iteratively varied as follows.

generator_lr= [0.0001,0.0002,0.0003,0.0004,0.0005]
discriminator_lr= [0.0001,0.0002,0.0003,0.0004,0.0005]
noise_multiplier= [0.0001,0.0005,0.001,0.002,0.003]
teacher_iters= [1, 3, 5, 7, 8] and
student_iters= [1, 3, 5, 7, 8]

In the final pass, the value of € was lowered to 2.5, the noise multiplier was set to 0.0001, and the student and teacher iterator was set to 2 to evaluate the impact of a reduced € value. After the generation of synthetic data, machine learning algorithms were trained to validate the quality and utility of the generated data in comparison with algorithms trained on real data. As the objective was to evaluate the quality of the generated data, not much emphasis was given to which ML model scored the best. The ROC scores for algorithms trained and tested on real data, trained and tested. on synthetic data, and finally trained on synthetic data and tested on real data are tabulated. The ROC Score is used as a measure to evaluate as compared to the accuracy score because the ROC score is better suited for evaluating the classifier algorithm trained using an imbalanced dataset.

V. RESULTS

The outcomes presented in this section are the summary of our results after narrowing down on the SDGs that are generating datasets that are close to real data based on their evaluation metrics. Table 2 tabulates the comparison of minority class percentages in the original dataset and the datasets generated using different generators.

We observe that not all the generators generate the minority target class when their population in the input sample is minuscule, as seen in the case of the Cerebral Stroke, Credit Card Application, and CredictCard Fraud datasets. Therefore, synthetic data generated by Gaussian, Copula, TVAE, DPGAN, DPCTGAN and PAC becomes unusable for machine learning training. Table 3 and Table 4 along with the figure in APPENDIX provides details of the machine learning model scores for all datasets generated using different parameters for synthetic data generators. The first observation is that, except for CTGAN, PATECTGAN, and PATEGAN, the rest of the generators do not generate the minority target class for all datasets. The Fig 12 shows the ROC curve plotting for all the generators using cerebral stroke as reference dataset. For cerebral stroke dataset, Gausian, DPGAN and DPCTGAN fail to produce the target class which is minority. Hence, they don't show results. TVAE, PAC and PATEGAN display marginal overfit.

The general usefulness of synthetic data is assessed using various metrics, including the confidence interval overlap, correlation mapping, relative entropy, KL Divergence, and Hellinger Distance.

Although PATEGAN generates all the target classes irrespective of the majority and minority split, it fails to produce synthetic data that matches the real data used in the experiments. The utility metrics used in the generic utility assessment of the three generators for the sample sizing of 2000 are presented in Table 5.

KL Divergence and the Euclidean Distance of PATEGAN is higher in comparison with PATECTGAN and CTGAN. Fig 4 shows the propensity distribution [16] comparison for the Logistic Classifier model trained on real data (blue color) in comparison with the propensity when trained and tested with synthetic data represented in green color and the distribution for model trained on synthetic data and tested on real data. From this graph it is evident that PATEGAN loses utility. This result is apparent even for other datasets used in our experiments.

DS Correlated also produces good results when the differential privacy $\in$ is high. However, it struggles to generate the data when the dimensionality of data is high as observed in the case of malware and wafer anomaly datasets. Therefore, we focus on CTGAN and PATECTGAN. The second inquiry pertains to the effect of increasing the sampling volume on the data generation. Fig 5 represents the ML model score comparison for different sampling sizes on CTGAN and PATECTGAN generators. The parameter chosen was epoch $= 300$ for CTGAN and $\in = 5$ for PATECTGAN. $\in$ is set to 5 to achieve better privacy scores.

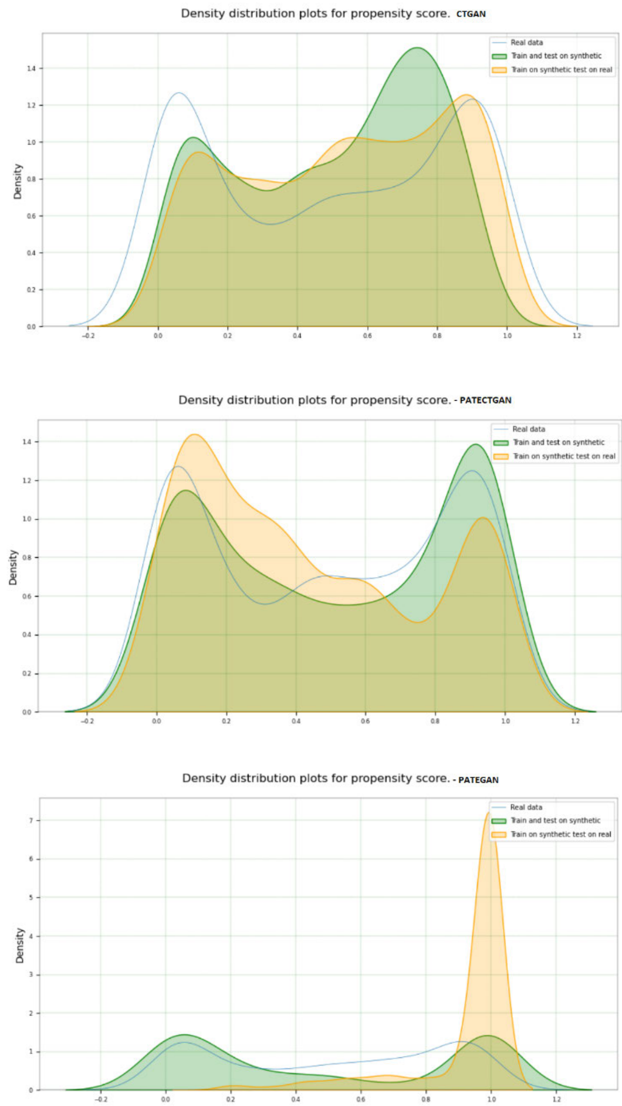
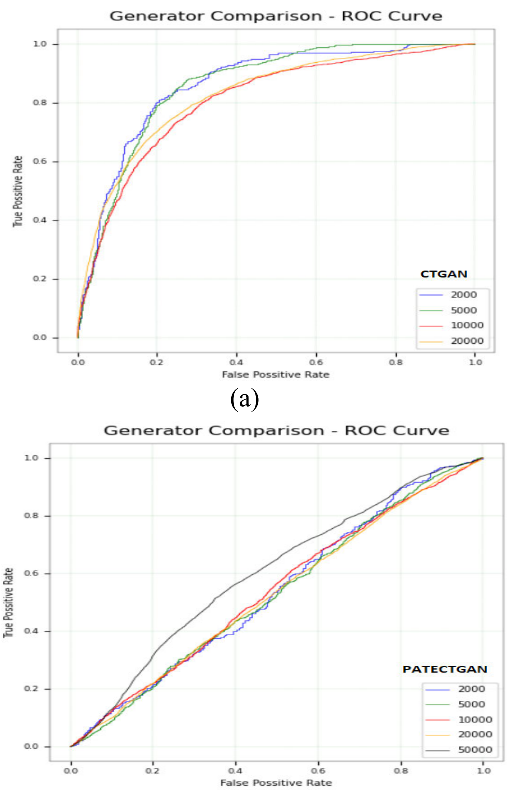


FIGURE 4. Comparison of propensity distribution of ML model for real data (blue), synthetic data (green) and model trained on synthetic data and tested on real data for datasets generated using CTGAN, PATECTGAN and PATEGAN generators.

As the $\in$ value is increased, PATEGAN produces better ROC scores which are tabulated in tables 3 and 4. The outcomes of the examination, wherein the sampling was iteratively increased from 2000 to 50000 rows for CTGAN and PATECTGAN generators using cerebral stroke data are provided in Table 6. The only notable observation is the privacy in CTGAN decreases as the number of sampling



(a)

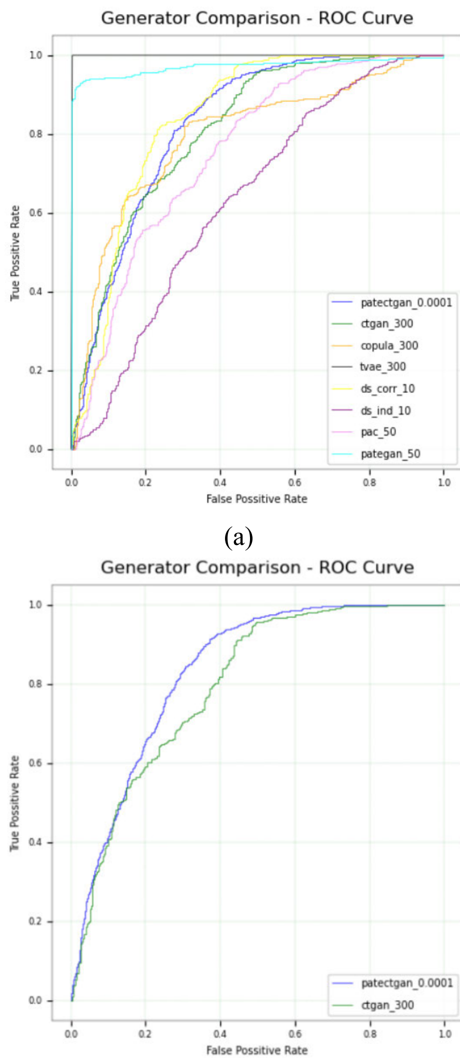
(b)

FIGURE 5. **Score for the model trained on synthetic data generated using different sample sizing for (a) CTGAN and (b) PATECTGAN.**

size is increased. Tables 7 and 8 tabulates outcomes of the examination, for the sampling size 50000 rows for all datasets. There are marginal variations; however, it is not very significant.

The next step in our assessment determines whether oversampling the minority class in the input sample results in a superior synthetic dataset. The SDV-based generators exhibited better results with oversampling of the minority class. Tables 9 and 10 show that TVAE generates the best machine learning score among the generators; however, its generic utility is inconsistent across datasets.

After verifying oversampling, we move towards which SDG produces synthetic data with the right balance across the evaluation metrics and what parameter gives the best result. Tables 3 and 4 tabulate the list of generators and the key parameters that influence the outcome of the generators. In the first iteration, *epoch* and *batch_size* for SDV, privacy budget € for DS, and SN-synth were used. In general, the ROC scores improved for all three packages (i.e., SDV, DS, and SN-synth) as the key parameter values increased. The CTGAN produced a better score as the epoch and batch-size values increased. A similar trend is observed for PATECTGAN and PATEGAN; the score continues to improve as the value of € increases from 1 to 50. Based on the scores for data generated using CTGAN and PATECTGAN, the final comparison is between these two generators. For closer comparison, the ROC scores of the logistic classifier model are tabulated in Tables 11 and 12, respectively. Logistic Regression showed greater variations than the other classifier algorithms. The comparison of the ROC scores for each of the generators for cerebral stroke data is illustrated in Fig 6.

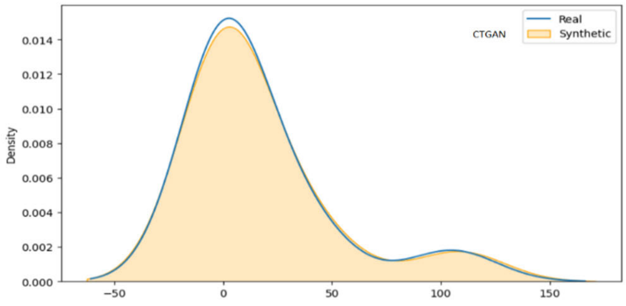


(a)

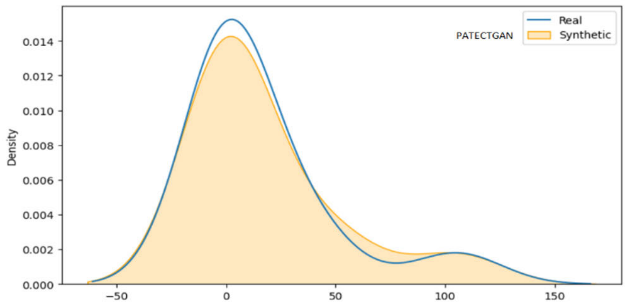
(b)

FIGURE 6. **(a). Comparison of the ROC scores of all the generators with the parameters that produced the best scores. (b). Comparison of the ROC curve for the data generated using the PATECTGAN with tuned parameter and CTGAN with epoch set to 300.**

The final step is to make variations to the parameters of PATECTGAN generator as described in previous section. Fig 7 illustrates the density overlap of real data and synthetic data generated using CTGAN [*epoch, batch_size*] = [300, 500] and PATECTGN with hyper-parameter tuning settings are *epoch = 300, batch_size= 500, € = 5, generator_lr= [0.0002], discriminator_lr= [0.0002], noise_multiplier= [0.0001], teacher_iters= [1], student_iters= [1],* dataset used is Cerebral Stroke. These parameters produced the best results.



(a)



(b)

FIGURE 7. **The density overlaps of real data and synthetic data generated using (a) CTGAN and (b) PATECTGAN with tuning.**

The correlation map comparison shown in Fig 8 and the other generic utility parameter comparison are provided in Tables 15 and 16 for CTGAN and PATECTGAN respectively.

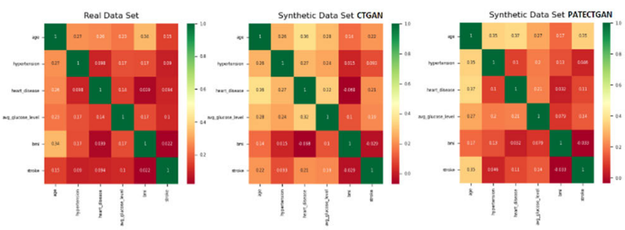


FIGURE 8. **Correlation map comparison of real data (Cerebral stroke dataset), synthetic data generated using CTGAN and PATECTGAN with tuned hyperparameters.**

Data Synthesizer also produces better scores for classification and regression models as the value of € is increased. PAC was the poorest of all the generators, given the type of datasets that were used. PAC introduced null values and constant values in the columns resulting in loss of data utility. It also fails to generate a minority class in the dataset.

The TVAE generator from the SDV package produces excellent results when the target class has a balanced distribution of categorical values; however, when the gap between the majority and minority data is large, TVAE fails to generate the minority target class. TVAE also fails to generate other minority groups in the dataset which is evident in the

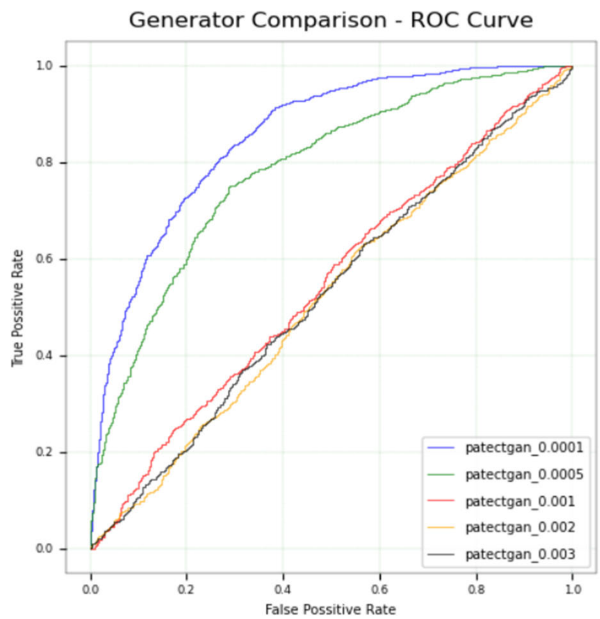


FIGURE 9. **Comparison of Logistic Regression classification score PATECTGAN with different noise_multipler values.**

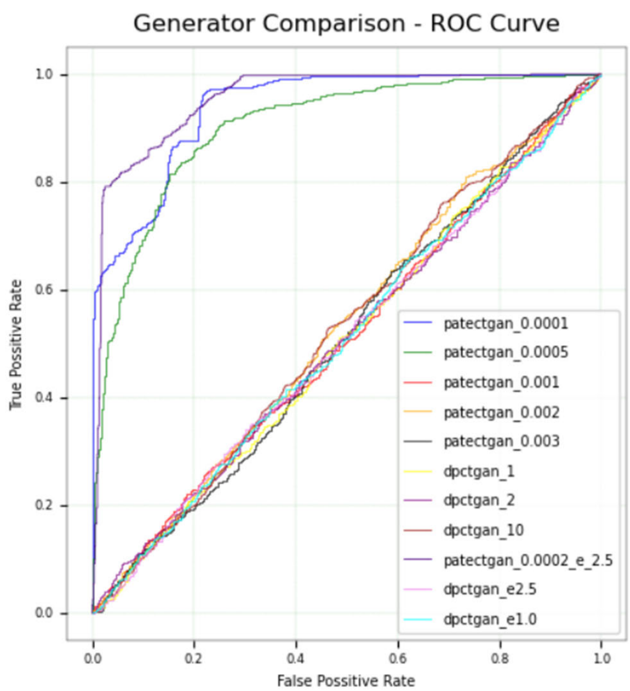


FIGURE 10. **Comparison of Logistic Regression classification model scores for PATECTGAN and DPCTGAN with different values of*noise_multplier* and sigma.**

mode graphs and fairness test using dalex. This behavior is very similar to that of DS. PATEGAN works in the opposite direction of other generators while dealing with the minority category of the target class. This produces a higher number of records with the target class which is a minority in the original dataset as shown in Table 2

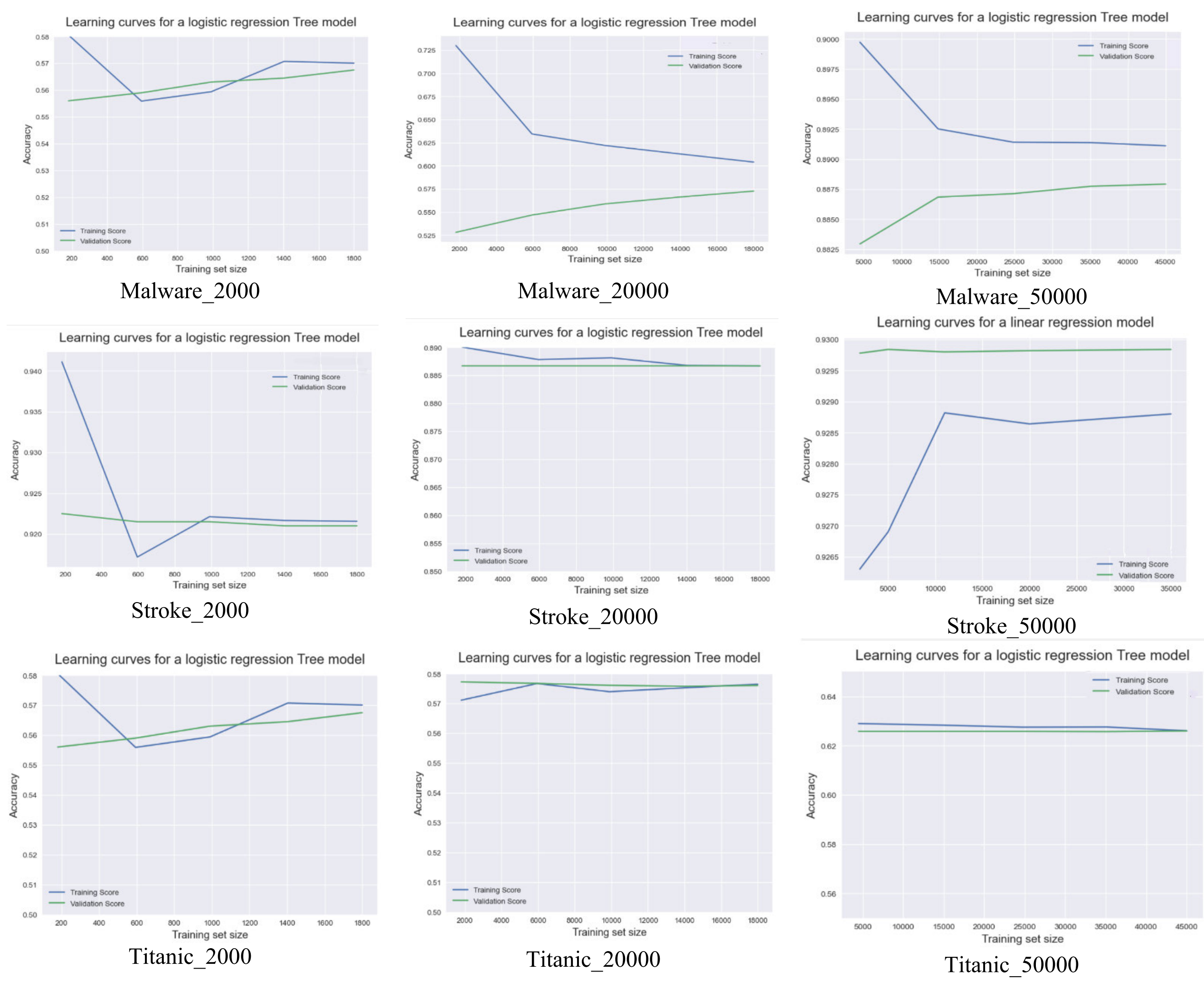


FIGURE 11. The Learning curve of logistic regression classifier model used on the synthetic data generated for three different datasets with different output sampling size. The curve shows that the accuracy flattens for the datasets around 15000 records.

TABLE 1. Datasets used as sample for synthetic data generation.

#	Dataset Name	# of rows	# of columns	Major Class	Minor Class	Ratio of major to minor	Target type
1	Detecting Anomalies in Wafer Manufacturing	2520	1559	1620	143	8.11%	Categorical
2	Malware Executable Detection	374	532	301	72	19.30%	Categorical
3	Titanic	891	12	549	342	38.38%	Categorical
4	Stroke Dataset	43401	12	42617	783	1.80%	Categorical
5	Cervical cancer	858	36	803	55	6.41%	Categorical
6	Adult census	48841	15	37154	11688	23.93%	Categorical
7	Smoke Detector	62631	15	44758	17847	28.51%	Categorical
8	HR Analysis	21287	14	14381	4777	24.93%	Categorical
9	Pima Indians Diabetes Database	769	9	500	268	34.90%	Categorical
10	Credit Card Approval Prediction (Cleaned Version)	25129	21	25007	122	0.49%	Categorical
11	Credit Card Fraud	284808	31	284316	492	0.17%	Categorical
12	Insurance Premium Data	1339	7	1259	80	5.97%	Continuous
13	House Rent Prediction	4747	12	4647	100	2.15%	Continuous

VI. DISCUSSION

How do the data generators treat the majority and minority classes in input data samples which have different dimensionality and datatypes (continuous, categorical, constants, and discrete)?

From the data tabulated in Table 1, we draw the following inference. CTGAN, PATECTGAN, and PATEGAN are the three SDGs that produced all categorical classes in the dataset for all types of input samples used in our experiments. DS independent and PAC-introduced constants.

TABLE 2. Percentage of minority target class in each of the dataset comparing the real dataset and the data generated using various generators. parameters column shows the key parameter that was used in the generation during one of the iterations.

	Parameters	Adult Income	Cervical Cancer	Cerebral Stroke	Credit Card Application	Creditcard Fraud	Diabetes	HRA	Malware	Smoke Detection	Titanic	Wafer Anomaly
Real Data		**24.95%**	**6.41%**	**1.7%**	**0.3%**	**0.1%**	**34.9%**	**25.25%**	**19.3%**	**28.54%**	**38.38%**	**7.15%**
Gaussian	epoch = 250	48.85%	5.00%	0.00%	0.00%	0.00%	42.65%	38.55%	10.50%	43.80%	45.90%	4.30%
CTGAN	epoch = 250	51.50%	14.10%	8.05%	5.40%	53.70%	46.55%	34.70%	48.05%	46.600%	30.05%	5.05%
Copula	epoch = 250	53.50%	2.90%	2.15%	0.0%	0.0%	54.35%	33.00%	27.55%	28.4500%	56.25%	9.00%
TVAE	epoch = 250	47.95%	0.0%	0.25%	0.0%	0.0%	26.30%	13.65%	22.20%	30.55%	33.40%	0.5%
DS Correlated	epsilon = 10	50.10%	10.35%	2.50%	0.35%	3.45%	45.80%	26.50%	DNG*	28.85%	35.60%	DNG*
DS Independent	epsilon = 10	48.85%	6.95%	2.05%	0.25%	0.20%	35.55%	24.95%	25.95%	28.45%	39.9%	0.00%
DPCTGAN	epsilon = 50	10.15%	0.05%	0.00%	0.00%	0.00%	61.90%	15.20%	0.25%	17.00%	34.70%	0.00%
DPGAN	epsilon = 50	0.00%	0.70%	0.00%	0.00%	7.60%	0.10%	45.55%	0.00%	0.00%	0.00%	0.00%
PAC	epsilon = 50	45.90%	0.00%	1.55%	0.00%	0.00%	22.80%	24.20%	0.00%	23.40%	45.95%	0.00%
PATECTGAN	epsilon = 50	57.35%	8.05%	14.25%	0.30%	0.60%	21.45%	42.00%	41.75%	7.30%	76.60%	2.30%
PATEGAN	epsilon = 50	46.40%	49.60%	99.70%	51.35%	0.70%	43.85%	38.35%	36.50%	14.45%	50.35%	95.60%

*DNG Did not generate due to resource constraints

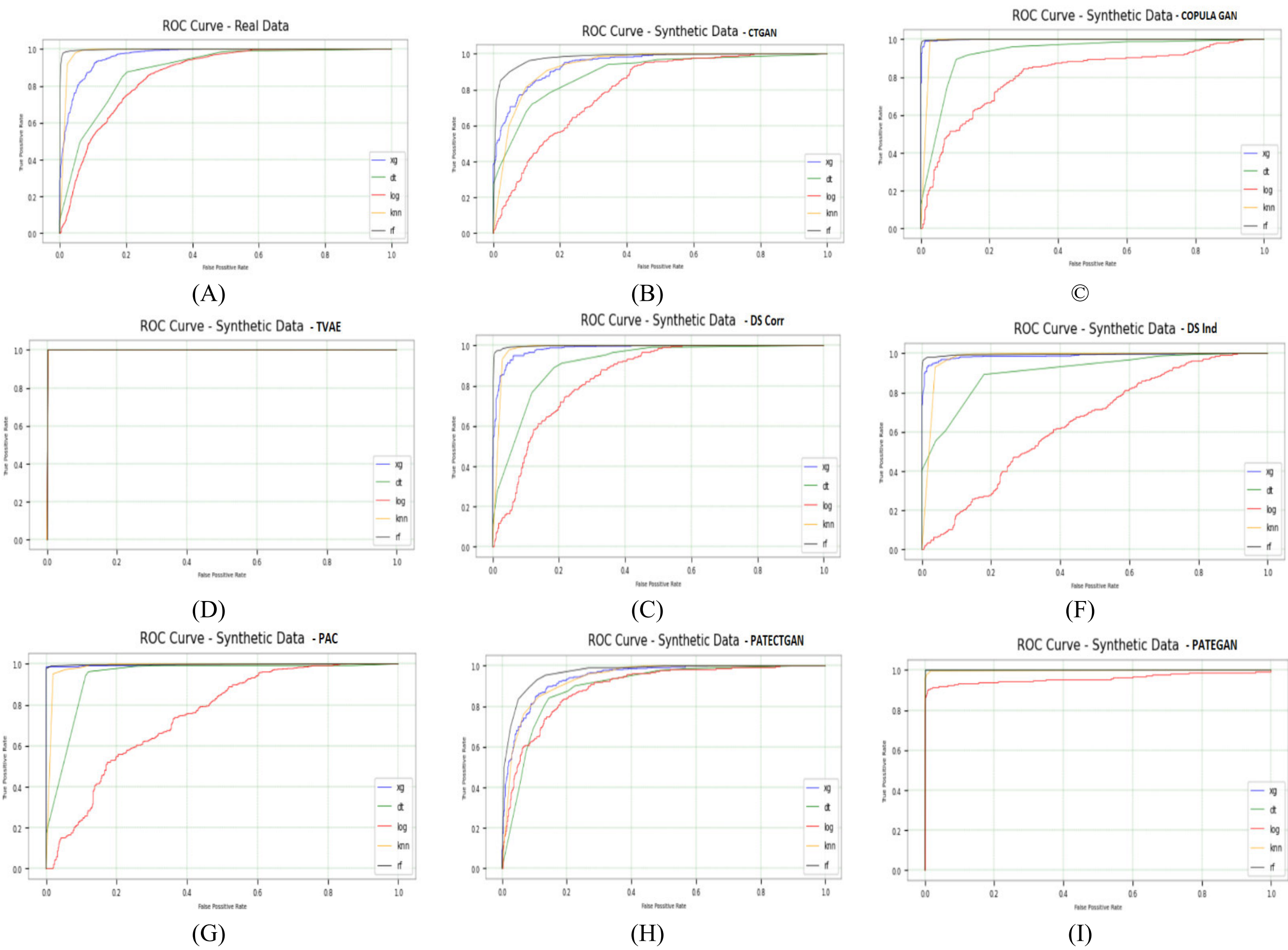


FIGURE 12. ROC Curve for Logistic Classifier model trained and tested on synthetic data generated using different generators for Cerebral Stroke dataset. Gaussian, DPGAN and DPCTGAN generators did not produce the target class which was a minority.

PAC also introduced null values that affected the utility of the dataset. DPCTGAN, DPGAN, and TVAE struggled to produce minority target classes, which made the dataset unusable for classification where the target class was imbalanced.

Does increasing the number of samples during data generation affect the quality of generated dataset?

This makes a difference for all the four SDV-based generators. For DS and SN-synth-based generators, increasing the sample size did not make much difference.

TABLE 3. Model trained and tested on synthetic data.

	Parameters	Adult Income	Cervical Cancer	Cerebral Stroke	Credit Card Application	Creditcard Fraud	Diabetes	HRA	Malware	Smoke Detection	Titanic	Wafer Anomaly
Original Data		**0.8203**	**0.9648**	**0.7649**	**0.9999**	**0.9511**	**0.8665**	**0.6268**	**0.9977**	**0.9727**	**0.4580**	**0.9451**
Gaussian	epoch = 10	0.7511	0.8816				0.7064	0.5513	0.9324	0.8530	0.7312	0.7565
CTGAN	epoch = 10	0.8675	0.6031	0.7732	0.8359	0.9665	0.6046	0.6483	0.9237	0.9823	0.5354	0.8873
Copula	epoch = 10	0.8567	0.9218	0.7881			0.5523	0.6781	0.5225	0.9449	0.5207	0.9460
TVAE	epoch = 10	0.8803					0.9614	0.8572		0.9674		
DS Correlated	epsilon = 0.1	0.5433	0.5838	0.6626	0.8654		0.5148	0.5433	0.4920	0.5447	0.5280	0.5245
DS Independent	epsilon = 0.1	0.6119	0.9382	0.7732	0.9206	0.7000	0.5184	0.6453		0.6876	0.5803	
DPCTGAN	epsilon = 1	0.5320	0.5299		0.5848	0.5970	0.5428	0.5875	0.5576	0.5787	0.5057	0.4878
DPGAN	epsilon = 1	0.8809	0.6807		0.9982	0.9329	0.6170	0.8275	0.9560	0.9222	0.6612	0.9747
PAC	epsilon = 1	0.6177		0.8245	0.9123			0.5609		0.8839	0.7980	
PATECTGAN	epsilon = 1	0.5234	0.5603	0.5056	0.5394	0.5672	0.5242	0.5130	0.6117	0.5228	0.5207	0.5139
PATEGAN	epsilon = 1	0.6391	0.7667	0.7078	0.6822	0.7972	0.6206	0.7544	0.9847	0.7358	0.5622	0.9904
Gaussian	epoch = 50	0.6794	0.8341				0.7269	0.5501	0.9535	0.8904	0.6877	0.7981
CTGAN	epoch = 50	0.8059	0.5897	0.7790	0.7795	0.9573	0.7457	0.7102	0.7154	0.9794	0.5481	0.9175
Copula	epoch = 50	0.8895	0.8761	0.8096			0.5439	0.6804	0.8265	0.9600	0.5112	0.9298
TVAE	epoch = 50	0.7657					0.9219	0.7803	0.9961	0.9505	0.9628	
DS Correlated	epsilon = 0.5	0.5544	0.5671	0.6457	0.9565	0.9983	0.5025	0.4752		0.5384	0.5281	
DS Independent	epsilon = 0.5	0.7301	0.9126	0.8643	0.9103	0.6626	0.5602	0.6033		0.8786	0.4669	
DPCTGAN	epsilon = 5	0.9166	0.5555	0.9532			0.5168	0.8256	0.9026	0.6540	0.5393	0.9990
DPGAN	epsilon = 5					1.0000	0.7942	0.9131	0.9313	0.9332	0.6231	0.9938
PAC	epsilon = 5	0.6308		0.8652	0.8985			0.5767		0.9407	0.6803	
PATECTGAN	epsilon = 5	0.5453	0.5676	0.5706	0.5920	0.5926	0.5798	0.4966	0.6350	0.5374	0.4835	0.9383
PATEGAN	epsilon = 5	0.6769	0.8048	0.5640	0.7123	0.8314	0.5975	0.6030	0.9739	0.7329	0.6326	0.9843
Gaussian	epoch = 100	0.6680	0.8713				0.7247	0.5474	0.9612	0.8852	0.7076	0.8015
CTGAN	epoch = 100	0.8516	0.5537	0.8186	0.7006	0.9652	0.6944	0.6947	0.8347	0.9777	0.5669	0.9146
Copula	epoch = 100	0.7958	0.7963	0.8477			0.5976	0.6414	0.8330	0.9534	0.5217	0.9669
TVAE	epoch = 100	0.7999		1.0000			0.9236	0.7914	0.9688	0.9519	0.9206	
DS Correlated	epsilon = 1	0.5373	0.5610	0.6260	0.9387	1.0000	0.5173	0.5276		0.5746	0.5273	
DS Independent	epsilon = 1	0.6186	0.9054	0.8257	0.9702	0.8410	0.5144	0.5855		0.8980	0.6020	
DPCTGAN	epsilon = 10	0.9564	0.7837	0.8527			0.5981	0.8492		0.9900	0.5030	0.6497
DPGAN	epsilon = 10			0.8292			0.8688		0.9840		0.7350	
PAC	epsilon = 10	0.6389		0.7855	0.9318			0.5608		0.9277	0.6261	
PATECTGAN	epsilon = 10	0.5621	0.7120	0.8849	0.9538		0.6814	0.5183	0.8967	0.7719	0.8506	0.7412
PATEGAN	epsilon = 10	0.6132	0.8421	0.6203	0.6312	0.7929	0.6865	0.8039	0.9529	0.7119	0.5527	0.9765
Gaussian	epoch = 200	0.7026	0.8125				0.7180	0.5375	0.9607	0.8478	0.7435	0.8340
CTGAN	epoch = 200	0.8116	0.6160	0.8643	0.9716	0.9720	0.7077	0.6785	0.8020	0.9916	0.5457	0.9232
Copula	epoch = 200	0.8373	0.6593	0.8581			0.5430	0.6752	0.7761	0.9657	0.4945	0.9392
TVAE	epoch = 200	0.7656					0.9570	0.8279	0.9945	0.9474	0.9517	0.9814
DS Correlated	epsilon = 5	0.5013	0.6215	0.6664	0.9687	1.0000	0.5278	0.5313	0.5913	0.5760	0.5205	
DS Independent	epsilon = 5	0.7582	0.9916	0.8709	0.9195	0.9258	0.5285	0.7067		0.9574	0.7404	
DPCTGAN	epsilon = 20	0.8685					0.6027	0.8432		0.9920	0.5404	
DPGAN	epsilon = 20						0.9032	0.9994				
PAC	epsilon = 20	0.6227		0.8139	0.8790			0.5547		0.9629	0.6775	
PATECTGAN	epsilon = 20	0.6175	0.8532	0.8155			0.8544	0.5166	0.9843	0.9911	0.8950	0.9491
PATEGAN	epsilon = 20	0.7521	0.7780	0.6241	0.8722	0.9024	0.6950	0.7785	0.9840	0.7576	0.6176	0.9911
Gaussian	epoch = 250	0.6940	0.8891				0.7462	0.5452	0.9449	0.8521	0.7639	0.7615
CTGAN	epoch = 250	0.8523	0.5711	0.8526	0.9565	0.9831	0.6977	0.6430	0.4697	0.9859	0.6042	0.9005
Copula	epoch = 250	0.8357	0.7503	0.8354			0.5181	0.7077	0.8716	0.9649	0.5083	0.8515
TVAE	epoch = 250	0.7696		1.0000			0.8615	0.7282	0.9930	0.9531	0.9147	0.9726
DS Correlated	epsilon = 10	0.5592	0.5668	0.6457	0.9768	1.0000	0.5204	0.5278	0.6695	0.5374	0.5118	
DS Independent	epsilon = 10	0.7938	0.9781	0.8468	0.9594	0.9586	0.5663	0.7074		0.9551	0.8416	
DPCTGAN	epsilon = 50	0.7790					0.5644	0.8204	0.9938	0.9966	0.5502	
DPGAN	epsilon = 50		0.7654			0.9951		0.9413			0.5873	
PAC	epsilon = 50	0.6397		0.7415				0.5413		0.9874	0.5726	
PATECTGAN	epsilon = 50	0.7529	0.9550	0.9071	0.9547	0.9325	0.8553	0.5295	0.9776	0.9914	0.7598	0.9558
PATEGAN	epsilon = 50	0.7550	0.8633	0.9711	0.7791	1.0000	0.7113	0.7379	0.9449	0.9361	0.7148	0.9854
Gaussian	epoch = 300	0.6879	0.8349				0.7564	0.5795	0.9148	0.8673	0.7156	0.7559
CTGAN	epoch = 300	0.8383	0.6486	0.8132	0.9931	0.9745	0.7103	0.7476	0.8322	0.9645	0.5947	0.9104
Copula	epoch = 300	0.7874	0.7871	0.8127			0.5296	0.6604	0.9069	0.9844	0.5520	0.9379
TVAE	epoch = 300	0.7876		1.0000			0.9739	0.7315	0.9967	0.9394	0.8854	

Does pre-processing (oversampling minority class) of the data before being fed to the generator have any impact?

Oversampling improves the classification score of machine learning models. The TVAE produces better datasets when minority classes are oversampled in the original dataset.

TABLE 4. Model trained on synthetic data and tested on real data.

	Parameters	Adult Income	Cervical Cancer	Cerebral Stroke	Credit Card Application	Creditcard Fraud	Diabetes	HRA	Malware	Smoke Detection	Titanic	Wafer Anomaly
Original Data		**0.8433**	**0.9756**	**0.8589**	**0.9995**	**0.9995**	**0.8544**	**0.7652**	**1.0000**	**1.0000**	**0.8730**	**0.9635**
Gaussian	epoch = 10	0.8198	0.9050				0.8648	0.7398	1.0000	0.9355	0.8659	0.8705
CTGAN	epoch = 10	0.7731	0.0920	0.6700	0.4134	0.9870	0.5033	0.6365	0.3742	0.9984	0.6514	0.4490
Copula	epoch = 10	0.7494	0.4184	0.5169			0.6259	0.6356	0.3913	0.9920	0.2769	0.4355
TVAE	epoch = 10	0.8156					0.8121	0.6164		0.9826		
DS Correlated	epsilon = 0.1	0.6412	0.5705	0.8328	0.6245	0.6987	0.4642	0.4718		0.8579	0.7828	0.4836
DS Independent	epsilon = 0.1	0.6771	0.3805	0.3575	0.7225		0.3790	0.4926	0.6798	0.8354	0.7675	0.5941
DPCTGAN	epsilon = 1	0.5185	0.5245		0.9725	0.7067	0.4577	0.5264	0.2622	0.9836	0.5669	0.5942
DPGAN	epsilon = 1	0.7528	0.5438		0.2721	0.4414	0.7264	0.6720	0.6870	0.0743	0.2998	
PAC	epsilon = 1	0.7965		0.7018			0.1190	0.5091		0.0302	0.6966	
PATECTGAN	epsilon = 1	0.7005	0.3229	0.5602	0.3264	0.0987	0.7564	0.5236	0.3460	0.6479	0.6499	0.4172
PATEGAN	epsilon = 1	0.3938	0.5505	0.4289	0.5296	0.4020	0.6643	0.3434	0.8491	0.6930	0.5631	0.4048
Gaussian	epoch = 50	0.8001	0.9179				0.8321	0.6892	0.9994	0.9554	0.8515	0.6729
CTGAN	epoch = 50	0.8054	0.1324	0.7511	0.7777	0.9959	0.6974	0.6156	0.8628	1.0000	0.4636	0.4861
Copula	epoch = 50	0.8109	0.0769	0.8009			0.6411	0.6267	0.7395	0.6761	0.3589	0.3410
TVAE	epoch = 50	0.8212					0.8053	0.5973	1.0000	0.9718	0.8184	
DS Correlated	epsilon = 0.5	0.7219	0.3501	0.8002	0.9121	0.7492	0.5008	0.5180		0.9936	0.5291	
DS Independent	epsilon = 0.5	0.5691	0.2187	0.3506	0.3489	0.4823	0.3655	0.4630		0.0220	0.7641	
DPCTGAN	epsilon = 5	0.7537	0.5467	0.6472		0.7492	0.4859	0.5769	0.5210	0.9996	0.4235	0.4663
DPGAN	epsilon = 5		0.3421			0.4492	0.3842	0.3871	0.8751	0.9405	0.7716	0.6077
PAC	epsilon = 5	0.7976		0.7675			0.2364	0.6966		0.9287	0.8249	
PATECTGAN	epsilon = 5	0.5406	0.6990	0.5732	0.0614	0.9379	0.4038	0.4155	0.9427	0.9918	0.4559	0.5546
PATEGAN	epsilon = 5	0.5045	0.6972	0.2570	0.2576	0.1325	0.6760	0.5259	0.4002	0.4924	0.2668	0.4890
Gaussian	epoch = 100	0.8070	0.9158				0.8899	0.6729	0.9999	0.9575	0.8406	0.7706
CTGAN	epoch = 100	0.8027	0.1911	0.7259	0.8385	0.9938	0.7643	0.6185	0.1966	0.9996	0.2066	0.3793
Copula	epoch = 100	0.7986	0.2613	0.7051			0.3227	0.5967	0.0178	0.8556	0.1959	0.4342
TVAE	epoch = 100	0.8158		0.5380			0.8040	0.5961	0.9983	0.9993	0.8523	
DS Correlated	epsilon = 1	0.6952	0.7521	0.7769	0.5247	0.9388	0.6640	0.5024		0.9969	0.5194	
DS Independent	epsilon = 1	0.4924	0.2104	0.3143	0.3821	0.3782	0.2969	0.5976		0.0714	0.6750	
DPCTGAN	epsilon = 10	0.6134	0.1907	0.5423			0.2659	0.6890		1.0000	0.7496	0.5206
DPGAN	epsilon = 10			0.7875			0.3616		0.7037		0.6230	
PAC	epsilon = 10	0.7973		0.5124			0.3827	0.5958		0.7740	0.7638	
PATECTGAN	epsilon = 10	0.5884	0.1913	0.2821	0.7032		0.8186	0.6260	0.9996	0.9859	0.8124	0.5457
PATEGAN	epsilon = 10	0.4016	0.2632	0.4344	0.6360	0.5750	0.6821	0.5752	0.5352	0.0098	0.6534	0.6610
Gaussian	epoch = 200	0.8090	0.9192				0.8557	0.7101	1.0000	0.9659	0.8404	0.8024
CTGAN	epoch = 200	0.8230	0.9192	0.7220	0.9936	0.9961	0.6821	0.6120	0.3381	0.9537	0.5494	0.6381
Copula	epoch = 200	0.7933	0.5078	0.8322			0.3361	0.6359	0.0016	0.9998	0.5818	0.4497
TVAE	epoch = 200	0.8127					0.8268	0.6230	1.0000	0.9999	0.7900	0.7947
DS Correlated	epsilon = 5	0.8000	0.9341	0.8419	0.4637	0.9962	0.7246	0.7418		0.9980	0.8112	
DS Independent	epsilon = 5	0.7361	0.1653	0.3369	0.2765	0.4720	0.4688	0.5136	0.4977	0.0267	0.8040	
DPCTGAN	epsilon = 20	0.7681					0.4196	0.6715		1.0000	0.7206	
DPGAN	epsilon = 20						0.3704	0.7005				
PAC	epsilon = 20	0.7792		0.8327			0.2666	0.6077		0.9150	0.8271	
PATECTGAN	epsilon = 20	0.6070	0.7537	0.7505	0.2813		0.8403	0.4330	1.0000	1.0000	0.8296	0.8404
PATEGAN	epsilon = 20	0.5333	0.1292	0.3705	0.5338	0.9386	0.3458	0.5321	0.1444	0.1266	0.6466	0.4535
Gaussian	epoch = 250	0.8176	0.9331				0.8226	0.7129	0.9967	0.9840	0.8451	0.7349
CTGAN	epoch = 250	0.8220	0.2180	0.8455	0.9761	0.9958	0.7969	0.6635	0.4293	0.9922	0.2775	0.5623
Copula	epoch = 250	0.8029	0.0980	0.7174			0.5627	0.7161	0.0000	0.9990	0.4078	0.3830
TVAE	epoch = 250	0.8096		0.5181			0.8394	0.6225	0.9993	0.9907	0.8135	0.6955
DS Correlated	epsilon = 10	0.8265	0.9260	0.8202	0.2566	0.3988	0.3997	0.4292		0.9998	0.8368	
DS Independent	epsilon = 10	0.6809	0.1031	0.2843	0.3499	0.4637	0.7315	0.7414	0.0128	0.0949	0.6542	
DPCTGAN	epsilon = 50	0.6223					0.5830	0.6801	0.4678	0.9975	0.7572	
DPGAN	epsilon = 50		0.4995			0.0474	0.2585	0.4775			0.6985	
PAC	epsilon = 50	0.7986		0.7692			0.3073	0.3860		0.9998	0.7810	
PATECTGAN	epsilon = 50	0.7682	0.9370	0.7493	*0.012346	*0.1303885	0.8409	0.6404	0.9900	0.9639	0.6370	0.6568
PATEGAN	epsilon = 50	0.5130	0.3361	0.6349	0.3956	0.1685	0.3915	0.4282	0.8459	0.9542	0.5966	0.5484
Gaussian	epoch = 300	0.8072	0.9404				0.8596	0.7041	0.9993	0.9421	0.8492	0.7196
CTGAN	epoch = 300	0.8201	0.2148	0.8229	0.9921	0.9916	0.5240	0.6541	0.1322	0.8764	0.8631	0.6007
Copula	epoch = 300	0.8168	0.2604	0.8266			0.3229	0.6094	0.0000	0.9940	0.6735	0.3523
TVAE	epoch = 300	0.8046		0.5787			0.8208	0.6211	1.0000	0.9930	0.7931	

* For PATEGAN generator, KNN classifier showed a better score for the Credit card datasets. Credit card application dataset had a score of **0.4388** and credit card fraud dataset got a score of **0.6028**.

Oversampled data are good for overcoming the challenges of privacy and bias, but we observe an inconsistency among the data sets in terms of data distribution and correlation. Therefore, this option works for scenarios if the only criterion

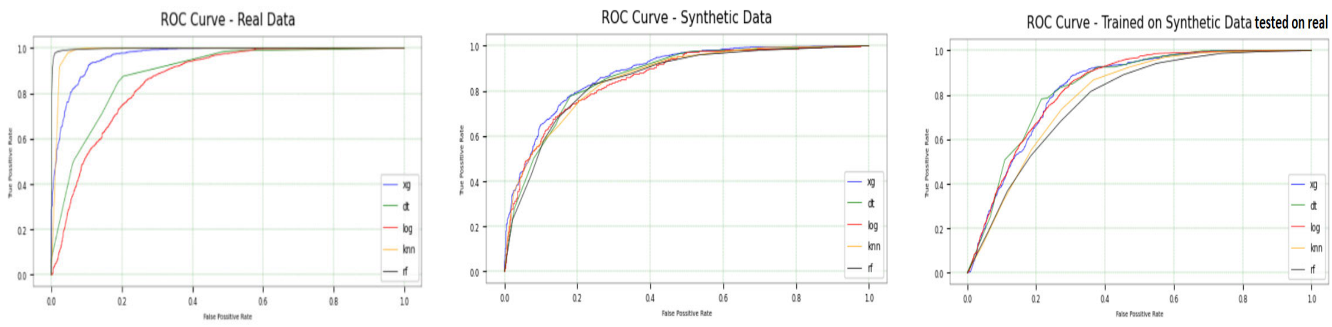


FIGURE 13. The ROC curve plotted for the cerebral stroke dataset when the data is oversampled for the target class and then generated using the TVAE generator.

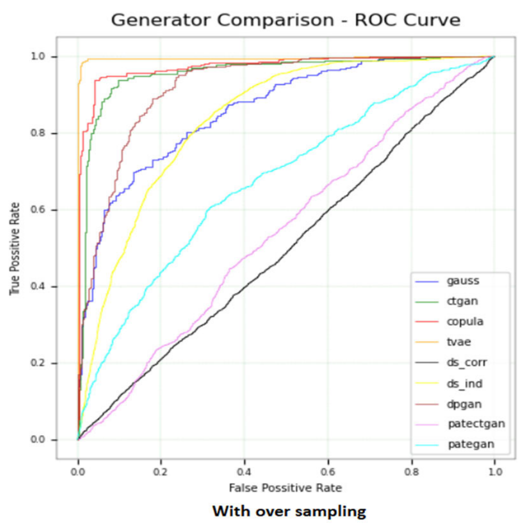


FIGURE 14. The ROC curve comparison for the datasets generated by different generators when the minority target class in the input dataset is oversampled and fed to synthetic data generator.

TABLE 5. Utility metrics comparison for ctgan, patectgan and pategan with 50000 samples for cerebral stroke data.

Generator	Type	age	avg_glucose_level	bmi
CTGAN	Hellinger	0.02702	0.0	6.7765e-16
	KL Div	44902.68	31215.25	3841.90
	Euclidean	1459.34	2847.24	487.01
PATECTGAN	Hellinger	0.009949	0.0	6.7764e-16
	KL Div	24637.69	29879.72	3555.69
	Euclidean	1362.88	2755.67	476.73
PATEGAN	Hellinger	0.00970	0.0	6.7764e-16
	KL Div	90339.43	226701.00	108157.63
	Euclidean	3933.68	10718.38	4291.59

is the model score, and data utility is not very important. Oversampling improves the classification score of machine learning models. The TVAE produces better datasets when minority classes are oversampled in the original dataset. Oversampled data are good for overcoming the challenges of privacy and bias, but we observe an inconsistency among the data sets in terms of data distribution and correlation. Therefore, this option works for scenarios if the only criterion is the model score and data utility is not very important.

Which synthetic data generator produces the dataset that matches or closely matches the real dataset in terms of its utility, disclosure control and fairness?

We utilize Table 2 as the foundational basis for our preliminary analysis. The data reveals that solely CTGAN, PATE-CTGAN, and PATEGAN generators possess the capability to generate the dataset for all the data samples. Subsequently, we proceed with the examination of the data enumerated in tables 3 through table 10. Upon careful observation of Fig 4, it becomes evident that PATEGAN fails to preserve the utility. Following an evaluation of the classification factors in Fig 6a and 6b, as well as the regression model scores in Table 19 and 20, it is apparent that the dataset generated by CTGAN and PATECTGAN exhibit promising results. To further validate these observations, we present the recorded data for CTGAN and PATECTGAN in Table 11 and 12 for comparative purposes. Ultimately, the Hellinger Distance, KL Divergence, Euclidean distance, and probability distributions are employed to assess the utility.Table 13 illustrates data privacy and Table 14, the metrics that exhibit bias for datasets generated using CTGAN and PATECTGAN. Based on these assessments, the CTGAN and PATECTGAN appeared to be the most suitable SDGs. It is worth noting that TVAE demonstrated optimally when the input data sample was well balanced.

What is the optimum parameter tuning required at the time of dataset generation?

For the SDV, it is recommended to use 300 epochs at the batch size of 500 to achieve superior results. The utility of the dataset is also enhanced under these conditions. However, the introduction of privacy measures may reduce the utility of the datasets. Unlike DS or SN-synth, SDV operates by utilizing the faker package of Python for the anonymization of data, rather than differential privacy. The DS's correlated

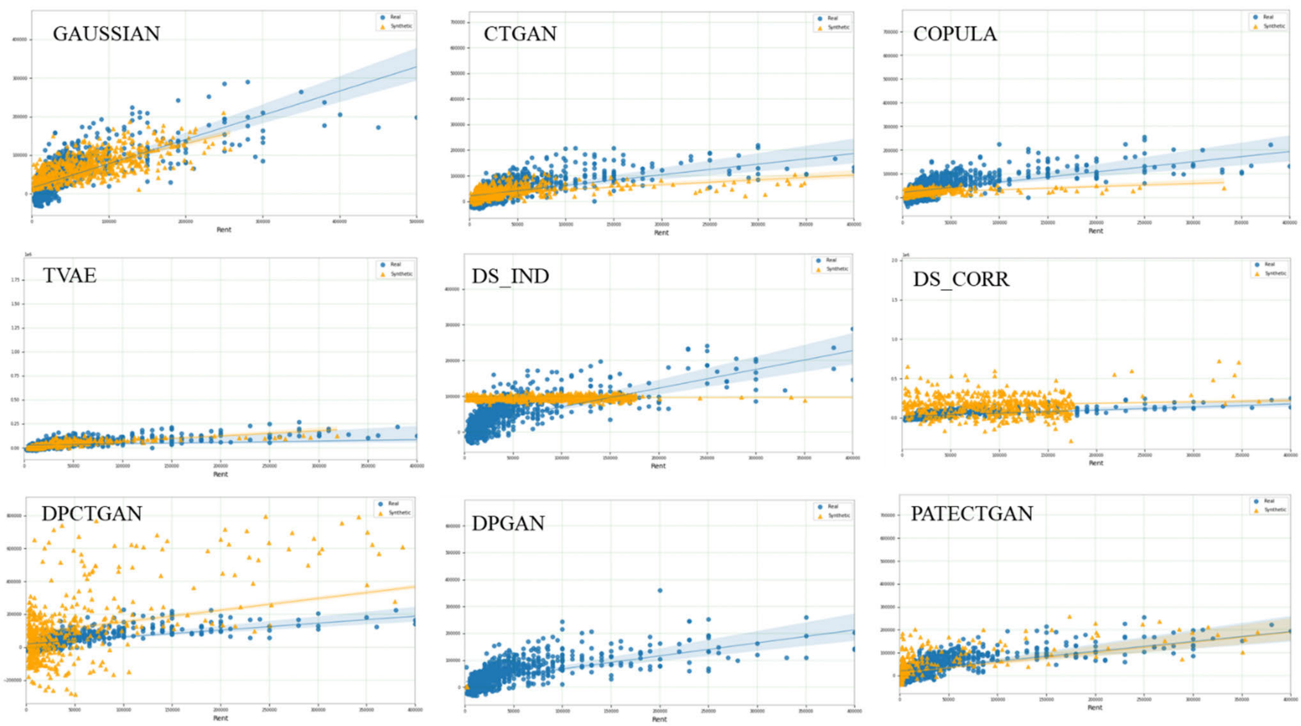


FIGURE 15. Regression plot comparison of House rent real data vs synthetic data for different synthetic data generators.

TABLE 6. ROC scores hellinger distance, entropy and privacy comparison for the data generated using pategan and CTGAN for different sampling size.

Generator	Sampling Size	ROC Score	Hellinger Distance (bmi)	entropy	Concept of Uniqueness (Privacy)
PATECTGAN	2000	0.5379	7.27E-16	0.085984	0.0
PATECTGAN	5000	0.5279	3.59E-16	0.064959	0.0
PATECTGAN	10000	0.5357	1.75E-16	0.083317	0.0
PATECTGAN	20000	0.5269	1.94E-16	0.091881	0.0
PATECTGAN	50000	0.5993	1.76E-16	0.056610	0.0
CTGAN	2000	0.8623	6.78E-16	0.059983	0.2232
CTGAN	5000	0.8593	3.70E-16	0.066367	0.3504
CTGAN	10000	0.8056	1.68E-16	0.063980	0.6554
CTGAN	20000	0.8259	2.97E-16	0.071601	1.00
CTGAN	50000	0.72061	6.10E-16	0.069674	1.00

TABLE 7. ROC scores for datasets generated by increasing the input sampling size to 50000. score when the model is trained and tested on synthetic data.

	Parameters	Adult Income	Cervical Cancer	Cerebral Stroke	Credit Card Application	Creditcard Fraud	Diabetes	HRA	Malware	Smoke Detection	Titanic	Wafer Anomaly
RealData		0.8437	0.9693	0.8560	0.9995	1.0000	0.8308	0.7419		1.0000	0.8277	0.9518
Gaussian	epoch = 300	0.7019	0.8323	0.9980			0.7481	0.5823	0.9340	0.8750	0.7263	0.7741
CTGAN	epoch = 300	0.8394	0.5608	0.7985	0.9591	0.9756	0.7188	0.6986	0.7662	0.9902	0.6053	0.9198
Copula	epoch = 300	0.8414	0.5597	0.8440			0.5199	0.6379	0.8556	0.9647	0.5055	0.8765
TVAE	epoch = 300	0.7864		0.9980			0.9453	0.7522	0.9876	0.9403	0.9575	
DS Correlated	epsilon = 0.5	0.7095	0.9082	0.8519	0.8948	0.6744	0.5138	0.5691		0.8544	0.5078	
DS Independent	epsilon = 0.5	0.4925	0.5589	0.5841	0.8024	0.8951	0.5006	0.5043		0.5031	0.5067	
DPCTGAN	epsilon = 5	0.8600	0.7027				0.5456	0.8713	0.9425	0.9951	0.5216	0.9838
DPGAN	epsilon = 5						0.7001	0.9954	0.9989		0.5955	
PAC	epsilon = 5	0.6801		0.9032			0.6680	0.5769		1.0000	0.8492	
PATECTGAN	epsilon = 5	0.5814	0.5849	0.6641	0.9580	0.9945	0.7416	0.5567	0.9404	0.7882	0.8691	0.8477
PATEGAN	epsilon = 5	0.6085	0.8677	0.6946	0.6876	0.8631	0.6026	0.6316		0.6668	0.6343	0.9879

generation option produces better scores for machine learning models as the privacy attribute € is increased. Nevertheless, that comes at the cost of compromising the privacy of datasets. DS-correlated is a high-resource

TABLE 8. ROC scores for classifier model trained on synthetic data and tested on original data.

	Parameters	Adult Income	Cervical Cancer	Cerebral Stroke	Credit Card Application	Creditcard Fraud	Diabetes	HRA	Malware	Smoke Detection	Titanic	Wafer Anomaly
RealData		0.8477	0.9938	0.8672	0.9988	1.0000	0.8530	0.7541	1.0000	1.0000	0.8494	0.9518
Gaussian	epoch = 300	0.8236	0.9120	0.6863			0.8336	0.6816	1.0000	0.9700	0.1939	0.8123
CTGAN	epoch = 300	0.8290	0.1443	0.7224	0.9906	0.9959	0.7968	0.6594	0.0422	0.9996	0.2720	0.2888
Copula	epoch = 300	0.8108	0.4037	0.8249			0.1871	0.6206	0.0000	0.9970	0.8059	0.4178
TVAE	epoch = 300	0.7789		0.4823			0.8337	0.6765	0.9991	0.9997	0.7598	
DS Correlated	epsilon = 0.5	0.7618	0.6812	0.8197	0.8850	0.8308	0.4261	0.3912		0.7357	0.1416	
DS Independent	epsilon = 0.5	0.5624	0.3497	0.3894	0.1730	0.2422	0.3806	0.4872		0.9980	0.7083	
DPCTGAN	epsilon = 5	0.6645	0.3958				0.4861	0.7015	0.0004	0.9973	0.2789	0.4717
DPGAN	epsilon = 5						0.3231	0.3245	0.8103		0.8564	
PAC	epsilon = 5	0.8087		0.8143			0.3900	0.6460		1.0000	0.8114	
PATECTGAN	epsilon = 5	0.5629	0.3936	0.4542	0.8106	0.0311	0.7987	0.6255	0.9996	0.7330	0.6536	0.6271
PATEGAN	epsilon = 5	0.6962	0.7042	0.5341	0.3253	0.3211	0.6804	0.4339		0.9481	0.8350	0.5476

TABLE 9. Logistic classifier model score for datasets trained and tested on synthetic data.

	Parameters	Adult Income	Cervical Cancer	Cerebral Stroke	Credit Card Application	Creditcard Fraud	Diabetes	HRA	Malware	Smoke Detection	Titanic	Wafer Anomaly
Original Data		0.8357	0.9732	0.8823	0.9999	0.9986	0.8568	0.7777	0.9972	0.9967	0.8377	0.9533
Gaussian	epoch= 250	0.7467	0.8479	0.7614	0.9590	0.8761	0.7535	0.6618	0.9865	0.8710	0.6991	0.7344
CTGAN	epoch= 250	0.8582	0.5272	0.8160	0.9601	0.9934	0.6737	0.8304	0.6751	0.9833	0.6678	0.9644
Copula	epoch= 250	0.8592	0.5179	0.8849	0.9355	0.9744	0.5481	0.7315	0.5161	0.9619	0.4954	0.6514
TVAE	epoch= 250	0.6877	0.9403	0.8399	0.9482	0.9625	0.9369	0.6648	0.9915	0.9405	0.8371	0.9220
DS Correlated	epsilon= .5	0.7293	0.6711	0.8395	0.6745	0.7640	0.4966	0.6010		0.8842	0.6490	
DS Independent	epsilon = 0.5	0.5073	0.5315	0.5193	0.4842	0.4961	0.5238	0.4713		0.4949	0.5210	
DPCTGAN	epsilon = 5	0.5217	0.5740				0.5029	0.8424	0.7682	0.8202	0.5000	0.7272
DPGAN	epsilon = 5	0.8766	0.8724	0.9228			0.6729	0.6839	0.9689	0.9945	0.6714	
PAC	epsilon = 5				0.8729		0.6856			0.9851	0.6189	
PATECTGAN	epsilon = 5	0.5236	0.5519	0.5232	*0.5749	*0.5778	0.5580	0.5532	0.7860	0.5148	0.5199	0.9527
PATEGAN	epsilon = 5	0.5918	0.8873	0.6816	0.7002	0.8186	0.6847	0.7290	0.9448	0.7186	0.5983	0.9846

For PATECTGAN generator, KNN and RandomForest classifier give a score of **0.9366** and **0.97613** respectively for credit card application data set. Similarly, **0.91540** and **0.96853** respectively for Credit Card Fraud dataset.

TABLE 10. Logistic classifier model score for datasets trained on synthetic data and tested on original data.

	Parameters	Adult Income	Cervical Cancer	Cerebral Stroke	Credit Card Application	Creditcard Fraud	Diabetes	HRA	Malware	Smoke Detection	Titanic	Wafer Anomaly
Original Data		0.8372	0.9779	0.8670	0.9999	0.9993	0.8799	0.7600	0.9972	0.9967	0.8651	0.9561
Gaussian	epoch = 250	0.8139	0.9461	0.8439	0.9917	0.9948	0.8787	0.6824	0.9899	0.9826	0.7520	0.8594
CTGAN	epoch = 250	0.8234	0.8399	0.8349	0.9940	0.9977	0.7594	0.6373	0.6890	0.9949	0.7184	0.6946
Copula	epoch = 250	0.7976	0.3684	0.8404	0.9941	0.9954	0.4462	0.5180	0.6587	0.9900	0.6297	0.3089
TVAE	epoch = 250	0.8060	0.9256	0.8377	0.9965	0.9979	0.8481	0.6068	1.0000	0.9665	0.7589	0.8869
DS Correlated	epsilon = 0.5	0.7364	0.4886	0.8404	0.9693	0.9952	0.4056	0.6064		0.9979	0.6750	
DS Independent	epsilon = 0.5	0.3962	0.1651	0.2978	0.1811	0.2055	0.5248	0.4831		0.0081	0.2908	
DPCTGAN	epsilon = 5	0.6177	0.4803				0.3964	0.6365	0.5010	0.9982	0.2612	0.3798
DPGAN	epsilon = 5	0.3487	0.7555	0.1723			0.3817	0.4318	0.8255	1.0000	0.4438	
PAC	epsilon = 5				0.6374		0.4655			0.9999	0.8290	
PATECTGAN	epsilon = 5	0.3983	0.5963	0.6956	0.4322	0.8395	0.7672	0.6336	0.9676	0.9995	0.6237	0.4323
PATEGAN	epsilon = 5	0.7549	0.1680	0.6005	0.4280	0.8360	0.3223	0.4220	0.2452	0.1209	0.6694	0.4638

consumer and struggles as data dimensionality increases, failing to generate datasets when the computing resources are limited. DS, DPGAN, and PATEGAN have limited optimization options, with varying privacy budget epsilon being the only feasible option. Therefore, PATECTGAN and DPCTGAN were selected for hyperparameter tuning. The optimal configuration for PATECTGAN involves setting *generator_lr, discriminator_lr, noise_multiplier, teacher_iters, and student_iters*, with a privacy budget of $\in = 5$. However, reducing the value of $\in$ while increasing the *noise_multiplier* can enhance disclosure control, but it may hurt the dataset utility. The quality of the generated

TABLE 11. The ROC Score of logistic regression classifier when the model is trained using synthetic data generated using CTGAN and tested against real data.

*Parameters	Adult Income	Cervical Cancer	Cerebral Stroke	Credit Card Application	Creditcard Fraud	Diabetes	HRA	Malware	Smoke Detection	Titanic	Wafer Anomaly
Original Data	**0.8433**	**0.9756**	**0.8589**	**0.9995**	**0.9995**	**0.8544**	**0.7652**	**1.0000**	**1.0000**	**0.8730**	**0.9635**
epoch = 10	0.7731	0.0920	0.6700	0.4134	0.9870	0.5033	0.6365	0.3742	0.9984	0.6514	0.4490
epoch = 50	0.8054	0.1324	0.7511	0.7777	0.9959	0.6974	0.6156	0.8628	1.0000	0.4636	0.4861
epoch = 100	0.8027	0.1911	0.7259	0.8385	0.9938	0.7643	0.6185	0.1966	0.9996	0.2066	0.3793
epoch = 200	0.8230	0.9192	0.7220	0.9936	0.9961	0.6821	0.6120	0.3381	0.9537	0.5494	0.6381
epoch = 250	0.8220	0.2180	0.8455	0.9761	0.9958	0.7969	0.6635	0.4293	0.9922	0.2775	0.5623
epoch = 300	0.8201	0.2148	0.8229	0.9921	0.9916	0.5240	0.6541	0.1322	0.8764	0.8631	0.6007

*Parameters *EPOCH* and BATCH_SIZE setting was set as [10, 50], [50, 50], [100, 100], [2000, 200], [250, 300], [300, 500] respectively

TABLE 12. The ROC score of logistic regression classifier when the model is trained using synthetic data generated using PATEGAN and tested against real data.

Parameters	Adult Income	Cervical Cancer	Cerebral Stroke	Credit Card Application	Creditcard Fraud	Diabetes	HRA	Malware	Smoke Detection	Titanic	Wafer Anomaly
epsilon = 1	0.7005	0.3229	0.5602	0.3264	0.0987	0.7564	0.5236	0.3460	0.6479	0.6499	0.4172
epsilon = 5	0.5406	0.6990	0.5732	0.0614	0.9379	0.4038	0.4155	0.9427	0.9918	0.4559	0.5546
epsilon = 10	0.5884	0.1913	0.2821	0.7032		0.8186	0.6260	0.9996	0.9859	0.8124	0.5457
epsilon = 20	0.6070	0.7537	0.7505	0.2813		0.8403	0.4330	1.0000	1.0000	0.8296	0.8404
epsilon = 50	0.7682	0.9370	0.7493	0.012	0.130	0.8409	0.6404	0.9900	0.9639	0.6370	0.6568
With *HPT	**0.7922**	**0.9598**	**0.8256**	**0.9822**	**0.9504**	**0.8441**	**0.7326**	**0.9911**	**0.9998**	**0.8480**	

*Hyper Parameter Tuning settings are *epoch* = 300, *batch_size*= 500, € = 5, generator_lr= [0.0002], discriminator_lr= [0.0002], *noise_multiplier*= [0.0001], teacher_iters= [1], student_iters= [1]

TABLE 13. Probability of identification of a record in a synthetic dataset using the concept of uniqueness.

Generator	**Adult Income**	**Cervical Cancer**	**Cerebral Stroke**	**Credit Card Application**	**Creditcard Fraud**	**Diabetes**	**HRA**	**Malware**	**Smoke Detection**	**Titanic**	**Wafer Anomaly**
GAUSSIAN	0.0000		0.0376	0.0000		0.0000	0.4810			0.0000	
CTGAN	0.4850		0.2291	0.0000		0.0000	0.4946			0.0000	
COPULA	0.4616		0.0764	0.0000		0.0000	0.4444			0.0000	
TVAE	0.4348		0.1010	0.0000		0.0000	0.6804			0.0000	
DS_CORR	0.0000		0.0000	0.0000		0.0000	0.0000			0.0000	
DS_IND	0.0000		0.0000	0.0000		0.0000	0.0000			0.0000	
DPCTGAN	0.4230		0.0000	0.0000		0.0000	0.0000			0.0000	
DPGAN	0.0000		0.0000	0.0000		0.0000	0.0000			0.0000	
PAC	0.0000		0.9570	0.0000		0.0000	0.0000			0.0000	
PATECTGAN	0.8680		0.0048	0.0000		0.0000	0.0000			0.0000	
PATEGAN	0.0000		0.0000	0.0000		0.0000	0.0000			0.0000	

The SDV generators lose privacy as the sampling size increases for Diabetes and Titanic datasets.

data is determined by a combination of these parameters along with *epoch and batch_size*.PATECTGAN utilizing *epoch* and *batch_size* values of 300 and 500 respectively and *generator_lr=0.0002, discriminator_lr=0.0002, noise_multiplier=0.0001, teacher_iters=1, and student_iters=1*achived the best result. As inferred from the graph in Fig 9, the optimal *noise_multiplier* value for PATECTGAN was 0.0001 at $\in = 5$. Other parameters did not make a major impact.

Further comparisons of PATECTGAN and DPCTGAN for various values of *noise_multiplier* and *sigma* respectively are shown in Fig 8. The comparison is run to validate whether varying the noise gradient indicator *sigma* for DPCTGAN has any impact on the generated dataset. However, from the graph, we infer that it does not provide superior scores for ML models or generic utility. From Fig 10, we also observe that PATECTGAN with $\in = 2.5$ and *noise_multipler* set to 0.0002 provides better ML model scores but it slightly loses its utility as compared with $\in = 5$.

Based on the evaluation and analysis, PATECTGAN with parameter tuning and CTGAN were the top two generators chosen for the final comparison. The disclo-

TABLE 14. A Comparison of original data and the sdgs ctgan and patectgan showing the number of metrics that has bias.

	Parameters	Adult Income	Cervical Cancer	Cerebral Stroke	Credit Card Application	Creditcard Fraud	Diabetes	HRA	Malware	Smoke Detection	Titanic	Wafer Anomaly
RealData		3		0	1		2	4			5	
Gaussian	epoch=250	0		0	DNG		3	3			5	
CTGAN	epoch=250	**1**		**0**	**1**		**3**	**2**			**3**	
Copula	epoch=250	1		0	DNG		4	3			0	
TVAE	epoch=250	DNG**		DNG	DNG		1	4			5	
DS Correlated	€ = 0.5	1		1	2		DNG	0			3	
DS_Independent	€ = 0.5	1		0	0		DNG	2			3	
DPCTGAN	€ = 5	DNG		DNG	0		1	3			0	
DPGAN	€ = 5	DNG		DNG	DNG		0	0			2	
PAC	€ = 5	0		DNG	DNG		DNG	3			5	
PATECTGAN	€ = 5	1		1	1		2	0			4	
PATECTGAN (HPT*)	€ = 5	**1**		**0**	**1**		**3**	**2**			**2**	
PATEGAN	€ = 5	1		1	2		3	0			4	

*Hyper Parameter Tuning settings are *epoch* = 300, *batch_size*= 500, € = 5, generator_lr= [0.0002], discriminator_lr= [0.0002], *noise_multiplier*= [0.0001], teacher_iters= [1], student_iters= [1]

DNG** Did Not Generate.

TABLE 15. A comparison of hellinger distance for different features of diabetes dataset generated using patectgan and ctagan generators.

	Pregnancies	Glucose	Blood Pressure	Skin Thickness	Insulin	BMI	Diabetes Pedigree Function	Age	Outcome
PATECTGAN	25.630146	42.78057	59.134121	65.383738	170.564626	23.747499	6.68663	25.319308	12.8062
CTGAN	27.894059	52.17175	44.479127	72.866798	195.143904	22.422535	8.08703	27.532011	13.6565

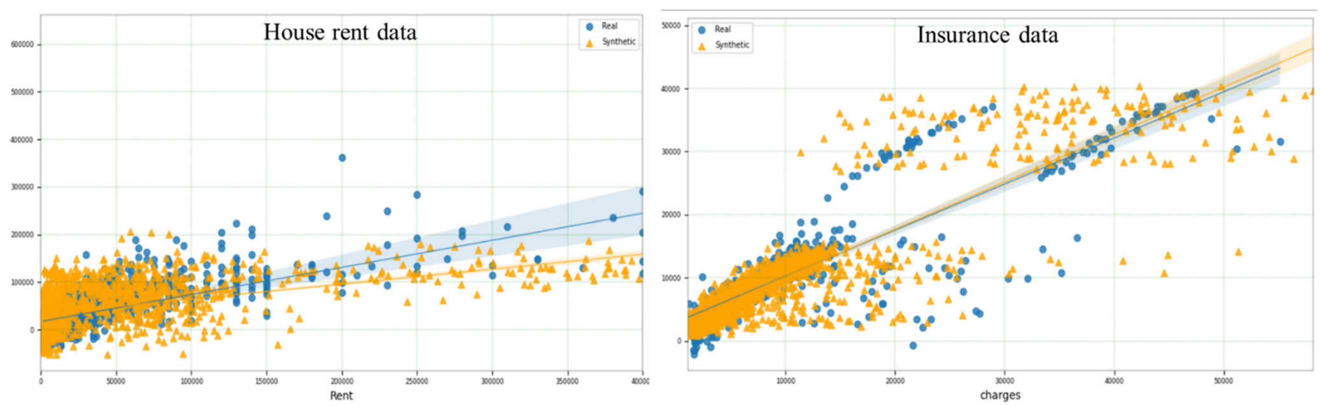


FIGURE 16. Regression plot comparison for House Rent data and Insurance data for the synthetic data generated using PATECTGAN with settings of *epoch* = 300,*batch_size*= 500, € = 5, generator_lr= [0.0002], discriminator_lr= [0.0002],*noise_multiplier*= [0.0001], teacher_iters= [1], student_iters= [1].

TABLE 16. A comparison of hellinger distance for different features of credit card approval dataset generated using patectgan and ctagan generators.

	Total_Income	Total_Good_Debt	Total_Bad_Debt
PATECTGAN	3.63E+01	72.836422	40.632378
CTGAN	2.50E+01	46.593603	29.903627

sure control between the two generators was compared by tabulating the data identification probability, as shown in Table 13. As expected, when privacy increases, the data utility deteriorates. CTGAN uses the "*anonymize_fields*" parameter to anonymize the data. When this feature was used to anonymize the fields of the adult census data, the utility of the generated data as well as the ROC score decreased.

The Python package provides five different metrics to assess the bias of machine learning classifiers. These metrics are the True Positive Rate (TPR), accuracy (ACC), Positive Predictive Value (PPV), False Positive Rate (FPR), and statistical parity (STP) which uses the minority subclass to

TABLE 17. Logistic classifier model.

	Adult Income	Cervical Cancer	Cerebral Stroke	Credit Card Application	Creditcard Fraud	Diabetes	HRA	Malware	Smoke Detection	Titanic	Wafer Anomaly
Real Data	**0.8274**	**0.9814**	**0.8660**	**0.9994**	**1.0000**	**0.8547**	**0.7592**	**1.0000**	**1.0000**	**0.8391**	**0.9459**
Train and test Synthetic data	0.7448	0.9866	0.8257	0.8846	0.9996	0.7671	0.6910	0.9977	0.9993	0.8945	0.9751
Train on synthetic data and test on real data	0.7839	0.9598	0.8342	0.9822	0.9504	0.8415	0.7326	0.9911	0.9998	0.8104	0.7593

generator_lr= [0.0002], discriminator_lr= [0.0002], noise_multiplier= [0.0001], teacher_iters= [1] and student_iters= [1]. pre-processing weightage *epsilon=5*.

TABLE 18. Random forest classifier model.

	Adult Income	Cervical Cancer	Cerebral Stroke	Credit Card Application	Creditcard Fraud	Diabetes	HRA	Malware	Smoke Detection	Titanic	Wafer Anomaly
Real Data	0.9248	0.9996	0.9967	1.0000	1.0000	0.8634	0.9007	1.0000	1.0000	0.8748	0.9459
Train and test Synthetic data	0.9623	0.9995	0.9375	0.9996	0.9985	0.9343	0.7967	0.9979	0.9996	0.9849	0.9944
Train on synthetic data and test on real data	0.7961	0.9655	0.7985	0.2467	0.8286	0.8145	0.6519	0.7308	0.9808	0.8164	0.7415

TABLE 19. House rent dataset.

Generator	Parameter	Train Score	Test Score	MSE	MAE	RMSE	R^2
Real Dataset		0.90442	0.13686	4240977100	20524.97	65122.78	0.14383
sdv_gauss	epoch = 250	0.88952	0.38517	1615677190	30188.7	40195.49	0.38599
sdv_ctgan	epoch = 250	0.84299	-0.07052	2097563614	26519.25	45799.17	-0.03147
sdv_copula	epoch = 250	0.77685	-0.17747	2114165024	21282.08	45980.05	-0.17704
sdv_tvae	epoch = 250	0.91919	0.558	712722109.2	10946.84	26696.86	0.55812
ds_ind	epsilon = 10	0.81082	-0.20256	5404574602	51055	73515.81	-0.20255
ds_corr	epsilon = 10	0.81445	0.13679	2.09771E+11	190982.23	458007.84	0.137
dpctgan	epsilon = 50	0.97354	0.84406	22544712338	79582.4	150148.97	0.84452
dpgan	epsilon = 50	1	1	0	0	0	nan
patectgan	epsilon = 50	0.91218	0.45556	4562448527	18387.64	67545.9	0.45557
pategan	epsilon = 50	0.87378	0.12852	10366408787	43813.02	101815.56	0.13083

determine bias. Table 14 shows the number of metrics that display bias in the dataset when used to train the Random Forest classifier model. To compare the data utility, the Hellinger distance for Diabetes and Credicard_Approval datasets were captured as samples and are shown in Tables 15 and 16, respectively. The data in these tables were captured using CTGAN parameters of *epoch*=250 and *batch_size*=300. and PATECTGAN parameters are *generator_lr= [0.0002], discriminator_lr= [0.0002], noise_multiplier= [0.0001], teacher_iters= [1], student_iters= [1]*.

From the data tabulated in Tables 19, 20, 21 and 22 for house rent and insurance data, PATECTGAN appears more superior. Fig 15 shows the regression plot comparison for real data and synthetic data whereas Fig 16 shows the comparison of house rent and insurance data generated using PATECTGAN with tuned parameters.

A. FUTURE WORK

TAVE is one of the best generators when the input sample data are balanced. It is also one of the less resource-intensive generators compared with all GAN-based generators. Therefore, we see scope to enhance the TVAE generator to identify and handle the data imbalance and bias internally during the generation, so that TVAE can be more reliable and widely used. Currently, there are methods available to balance data either through pre-processing or post-processing [31]. It will be worthwhile introducing the in-processing feature into TVAE so that it can become a trusted SDG. Understanding why

TABLE 20. Insurance dataset.

Generator	Parameter	Train Score	Test Score	MSE	MAE	RMSE	R^2
Real Dataset		0.96817	0.81277	27444317.99	2730.1	5238.73	0.81278
sdv_gauss	epoch = 250	0.86154	0.18565	90748907.61	7475.24	9526.22	0.18571
sdv_ctgan	epoch = 250	0.79636	-0.16468	130667761.5	8214.73	11431	-0.16429
sdv_copula	epoch = 250	0.78179	-0.14279	266252816.6	12899.16	16317.26	-0.1365
sdv_tvae	epoch = 250	0.92383	0.53168	55393565.98	4426.2	7442.69	0.53171
ds_ind	epsilon = 10	0.79858	-0.25281	173538912.5	10220.98	13173.42	-0.24081
ds_corr	epsilon = 10	0.87876	0.24785	196712852.9	9695.27	14025.44	0.24924
dpctgan	epsilon = 50	0.7994	-0.14878	194116649.5	11382.82	13932.58	-0.13307
dpgan	epsilon = 50	0.93307	0.60823	58323313.37	3655.36	7636.97	0.60823
patectgan	epsilon = 50	0.94386	0.66469	30693286.9	3637.71	5540.15	0.66552
pategan	epsilon = 50	0.84903	0.0396	43308094.26	4805.99	6580.89	0.04039

TABLE 21. METRIC for house rent dataset for dataset generated using patectgan with noise_multipler = 0.0001 and € = 5. captured scores are for random forest regressor model.

Parameter	Train Score	Test Score	MSE	MAE	RMSE	R^2
Original Data	0.91505	0.25569	9225093864	14764.58	96047.35	0.25569
Trained and tested on Synthetic	0.94119	0.68899	2310763597	25080.28	48070.4	0.68899
Trained on synthetic, tested on real	0.94287	0.14007	10658047917	24238.97	103237.82	0.14007

TABLE 22. Metric for insurance dataset for dataset generated using patectgan with noise_multipler = 0.0001 and € = 5. captured scores are for random forest regressor model.

Generator	Train Score	Test Score	MSE	MAE	RMSE	R^2
Original Data	0.97467	0.78261	30083095.72	2932.14	5484.81	0.78261
Trained and tested on Synthetic	0.94919	0.75392	26049815.55	3123.51	5103.9	0.75392
Trained on synthetic, tested on real	0.94937	0.7337	36850997.84	3769.4	6070.5	0.7337

many synthetic data generators other than PATECTGAN and CTGAN struggle to generate the minority target class.

VII. CONCLUSION

From our experiments we observe that no single SDG can handle all input sample scenarios perfectly. Therefore, our findings cannot be generalized. Based on results tabulated in Table 2, 3, 4 and further comparing the evaluation metrics We narrow down on PATECTGAN and CTGAN as the top two options. TVAE and DPGAN were found to introduce bias by dropping classes with small representation sizes. In both DS and SN-synth, increasing the value of the privacy weightage leads to a boost in the data utility. TVAE is the most effective in generating high-quality data for balanced datasets, even when minority classes are oversampled in the input data. While PATECTGAN, with the smallest noise multiplier of 0.0001 and € =2.5, yields better results for the classification model, a *noise_multiplier* of 0.0001 with € =5 is slightly better in terms of utility. It is important to note that increasing the noise multiplier for better privacy comes at the cost of data utility and introduces more bias into the dataset. CTGAN exhibits similar behavior. The commercial generator provided by GretelAI exhibited no variation. Consequently, the selection of privacy-enhancing parameters must be contingent on the requirements of the business. If the statistical attributes of the dataset and the accuracy of machine learning prevail as the primary business needs, then CTGAN without anonymization and with epoch and *batch_size* values set to maximum, or PATECTGAN with a *noise_multiplier* of 0.0001 with € =5 are the most optimal options.

REFERENCES

[1] D. B. Rubin, "Statistical disclosure limitation (SDL)," *J. Official Statist.*, vol. 9, no. 2, pp. 461–468, 1993, doi: 10.1007/978-0-387-39940-9_3686.

[2] T. E. Raghunathan, J. P. Rubin, and D. B. Reiter, "Multiple imputation for statistical disclosure limitation," *J. Off. Statist.*, vol. 19, no. 1, pp. 1–16, 2003. [Online]. Available: http://hbanaszak.mjr.uw.edu.pl/TempTxt/RaghunathanEtAl_2003_MultipleImputationforStatisticalDisclosureLimitation.pdf

[3] C. Dwork, "Differential privacy: A survey of results," in *Proc. 5th Int. Conf. Theory Appl. Models Comput.*, Lecture Notes in Computer Science, vol. 4978, 2008, pp. 1–19, doi: 10.1007/978-3-540-79228-4_1.

[4] H. Ping, J. Stoyanovich, and B. Howe, "DataSynthesizer: Privacy-preserving synthetic datasets," in *Proc. ACM Int. Conf. Proc. Ser.*, 2017, p. F1286, doi: 10.1145/3085504.3091117.

[5] M. Hittmeir, A. Ekelhart, and R. Mayer, "Utility and privacy assessments of synthetic data for regression tasks," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2019, pp. 5763–5772, doi: 10.1109/BigData47090.2019.9005476.

[6] M. Hittmeir, A. Ekelhart, and R. Mayer, "On the utility of synthetic data: An empirical evaluation on machine learning tasks," in *Proc. 14th Int. Conf. Availability, Rel. Secur.*, Aug. 2019, pp. 1–6, doi: 10.1145/3339252.3339281.

[7] F. K. Dankar, M. K. Ibrahim, and L. Ismail, "A multi-dimensional evaluation of synthetic data generators," *IEEE Access*, vol. 10, pp. 11147–11158, 2022, doi: 10.1109/ACCESS.2022.3144765.

[8] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, vol. 16, pp. 321–357, Jun. 2002, doi: 10.1613/jair.953.

[9] V. Cheng, V. M. Suriyakumar, N. Dullerud, S. Joshi, and M. Ghassemi, "Can you fake it until you make it? impacts of differentially private synthetic data on downstream classification fairness," in *Proc. ACM Conf. Fairness, Accountability, Transparency*, Mar. 2021, pp. 149–160, doi: 10.1145/3442188.3445879.

[10] G. Ganev, B. Oprisanu, and E. De Cristofaro, "Robin hood and Matthew effects: Differential privacy has disparate impact on synthetic data," 2021, *arXiv:2109.11429*.

[11] E. Bagdasaryan, O. Poursaeed, and V. Shmatikov, "Differential privacy has disparate impact on model accuracy," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 15479–15488.

[12] C. Tran, M. H. Dinh, and F. Fioretto, "Differentially empirical risk minimization under the fairness lens," 2021, *arXiv:2106.02674*.

[13] A. F. Karr, C. N. Kohnen, A. Oganian, J. P. Reiter, and A. P. Sanil, "A framework for evaluating the utility of data altered to protect confidentiality," *Amer. Stat.*, vol. 60, no. 3, pp. 224–232, 2006, doi: 10.1198/000313006X124640.

[14] J. P. Reiter, "Estimating risks of identification disclosure in microdata," *J. Amer. Stat. Assoc.*, vol. 100, no. 472, pp. 1103–1112, Dec. 2005, doi: 10.1198/016214505000000619.

[15] K. Bhanot, M. Qi, J. S. Erickson, I. Guyon, and K. P. Bennett, "The problem of fairness in synthetic healthcare data," *Entropy*, vol. 23, no. 9, p. 1165, Sep. 2021, doi: 10.3390/e23091165.

[16] F. K. Dankar and M. Ibrahim, "Fake it till you make it: Guidelines for effective synthetic data generation," *Appl. Sci.*, vol. 11, no. 5, p. 2158, Feb. 2021, doi: 10.3390/app11052158.

[17] J. Snoke, G. M. Raab, B. Nowok, C. Dibben, and A. Slavkovic, "General and specific utility measures for synthetic data," *J. Roy. Stat. Soc. Ser. A, Statist. Soc.*, vol. 181, no. 3, pp. 663–688, Jun. 2018, doi: 10.1111/rssa.12358.

[18] R. Heyburn, R. R. Bond, M. Black, M. Mulvenna, J. Wallace, D. Rankin, and B. Cleland, "Machine learning using synthetic and real data: Similarity of evaluation metrics for different healthcare datasets and for different algorithms," in *Proc. Data Sci. Knowl. Eng. Sens. Decis. Support*, Sep. 2018, pp. 1281–1291, doi: 10.1142/9789813273238_0160.

[19] S. I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, Y. B. Ozair, and A. Courville, "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 27, 2014, pp. 2672–2680.

[20] J. Jordon, J. Yoon, and M. Van Der Schaar, "PATE-GaN: Generating synthetic data with differential privacy guarantees," in *Proc. 7th Int. Conf. Learn. Represent.*, 2019, pp. 1–21.

[21] B. Vega-Márquez, C. Rubio-Escudero, and I. Nepomuceno-Chamorro, "Generation of synthetic data with conditional generative adversarial networks," *Log. J. IGPL*, vol. 30, no. 2, pp. 252–262, Mar. 2022, doi: 10.1093/jigpal/jzaa059.

[22] D. Xu, S. Yuan, L. Zhang, and X. Wu, "FairGAN: Fairness-aware generative adversarial networks," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2018, pp. 570–575, doi: 10.1109/BIGDATA.2018.8622525.

[23] L. Rosenblatt, X. Liu, S. Pouyanfar, E. de Leon, A. Desai, and J. Allen, "Differentially private synthetic data: Applied evaluations and enhancements," 2020, *arXiv:2011.05537*.

[24] L. Xie, K. Lin, S. Wang, F. Wang, and J. Zhou, "Differentially private generative adversarial network," 2018, *arXiv:1802.06739*.

[25] T. Farrand, F. Mireshghallah, S. Singh, and A. Trask, "Neither private nor fair: Impact of data imbalance on utility and fairness in differential privacy," in *Proc. Workshop Privacy-Preserving Mach. Learn. Pract.*, Nov. 2020, pp. 15–19, doi: 10.1145/3411501.3419419.

[26] B. Bullwinkel, K. Grabarz, L. Ke, S. Gong, C. Tanner, and J. Allen, "Evaluating the fairness impact of differentially private synthetic data," 2022, *arXiv:2205.04321*.

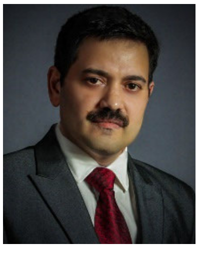
[27] S. Ghalebikesabi, H. Wilde, J. Jewson, A. Doucet, S. Vollmer, and C. Holmes, "Mitigating statistical bias within differentially private synthetic data," 2021, *arXiv:2108.10934*.

[28] N. Patki, R. Wedge, and K. Veeramachaneni, "The synthetic data vault," in *Proc. IEEE Int. Conf. Data Sci. Adv. Analytics (DSAA)*, Oct. 2016, pp. 399–410, doi: 10.1109/DSAA.2016.49.

[29] J. G. Bethlehem, W. J. Keller, and J. Pannekoek, "Disclosure control of microdata," *J. Amer. Stat. Assoc.*, vol. 85, no. 409, pp. 38–45, Mar. 1990, doi: 10.1080/01621459.1990.10475304.

[30] H. Baniecki, W. Kretowicz, P. Piatyszek, J. Wisniewski, and P. Biecek, "dalex: Responsible machine learning with interactive explainability and fairness in Python," *J. Mach. Learn. Res.*, vol. 22, pp. 1–7, Jan. 2021.

[31] E. Barbierato, M. L. D. Vedova, D. Tessera, D. Toti, and N. Vanoli, "A methodology for controlling bias and fairness in synthetic data generation," *Appl. Sci.*, vol. 12, no. 9, p. 4619, May 2022, doi: 10.3390/app12094619.



A. KIRAN was born in Puttur, Karnataka, India, in 1975. He received the B.E. degree in electronics and communication and the M.S. degree in software systems from the Birla Institute of Technology and Science, Pilani, Rajasthan, India, in 1997 and 2004, respectively. He is currently pursuing the Ph.D. degree.

He is the Associate Director in one of the prestigious American multinational companies in the field of artificial intelligence and automation. Being passionate about exploring, investigating, and learning, he enrolled for the Ph.D. degree, in 2021. His research interests include artificial intelligence, machine learning, and automation. As a result, the research topic is focused on synthetic data which in the present era where security, confidentiality, and data privacy are talked about, is seen as a capable substitute for real data for machine learning.

Mr. Kiran has presented two conference papers on this topic organized by Springer and the other one by IEEE.

S. SARAVANA KUMAR (Member, IEEE) received the Ph.D. degree from Bharath University, Chennai, India.

He is currently a Professor and the Head/IT of the AI and ML Programme, School of Engineering and Technology (SOET), CMR University, Bengaluru. After completing the Ph.D. degree, he took up his passion for research. In parallel, he began teaching. Having 20 years of experience in teaching and research, he has authored many books and filed 28 patents on various IoT sensor techniques. In his profession as an academician, he has guided 15 research scholars. He is also guiding research scholars associated with CMR University and with consultancy projects in AI using machine learning techniques. He has published more than 100 papers in various internationally reputed SCI journals and presented several papers at various national and international conferences.

Dr. Kumar is a member of the IEEE Computer Society Bangalore Chapter. He is also an editor and a reviewer of several international journals.