Tabular Data Synthesis with GANs for Adaptive AI Models

Sandeep Hans*
shans001@in.ibm.com
IBM Research, India

Anupam Sanghi*
anupam.sanghi1@ibm.com
IBM Research, India

Diptikalyan Saha
diptsaha@in.ibm.com
IBM Research, India

ABSTRACT

In situations such as demographics change ML models often perform poorly because the training data does not appropriately represent the environment. Privacy concerns worsen the issue by severely limiting training data. In this paper, we present a framework that utilizes a GAN-based synthesizer to generate synthetic data that not only satisfies user-defined constraints expressed as marginal distributions of selected columns but also strives to preserve the distributions observed in the original data. This framework takes as input an original dataset and a set of user-defined constraints, and synthesizes data that adheres to these constraints while capturing the underlying distributions present in the given data. The result is a customizable and realistic data generation solution that balances constraint satisfaction and preservation of data distributions.We validate and demonstrate the effectiveness of our technique through experimentation.

ACM Reference Format:
Sandeep Hans, Anupam Sanghi, and Diptikalyan Saha. 2024. Tabular Data Synthesis with GANs for Adaptive AI Models. In *7th Joint International Conference on Data Science & Management of Data (11th ACM IKDD CODS and 29th COMAD) (CODS-COMAD 2024), January 04–07, 2024, Bangalore, India.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3632410.3632438

1 INTRODUCTION

Artificial Intelligence (AI) systems are now being utilized in a wide range of applications, including medical diagnosis, autonomous vehicles, and financial trading. These critical domains rely on AI to provide accurate and timely insights. However, for the broader adoption of AI systems, it is essential to establish trust in their output and ensure their reliability.

As models often fail due to data distribution shifts in production, simulating anticipated data-drift conditions in testing is vital. This allows for the creation of synthetic payloads to assess the model's performance. Developing customized synthetic test data becomes imperative to meet these needs. Developing data based on user-specified customizations is crucial for tailoring synthetic data to specific requirements, including the configuration of marginal distributions to address distribution shifts and align with real-world scenarios. Tabular data synthesis presents key challenges for accurate synthetic data generation, including mixed data types (discrete and continuous distributions), non-Gaussian distributions, multimodal distributions, sparse one-hot encoding for categorical columns, and highly imbalanced categories.

*Both authors contributed equally to the paper

CODS-COMAD 2024, January 04–07, 2024, Bangalore, India
© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-1634-8/24/01…$15.00
https://doi.org/10.1145/3632410.3632438

Although existing techniques like GAN [9], Variational Autoencoder [10] can generate synthetic realistic data, they are not customizable by user-specification and do not allow external customizations. There could be data transformation such as change in demographics or futuristic scenarios where data distribution may change. For example, change in *Male* to *Female* ratio from $4:3$ to $1:1$ and *Salary* distribution from `Gaussian` to `uniform`. Another tabular synthesis framework, AITEST [18], proposes an automated test data synthesis framework for black-box classification models using tabular data. While it generates realistic synthetic data with user-controllable constraints, the associations mentioned do not capture complex column dependencies in general datasets, limiting their generalizability. Moreover, true functions in datasets can significantly differ from the functions mentioned under constraints.

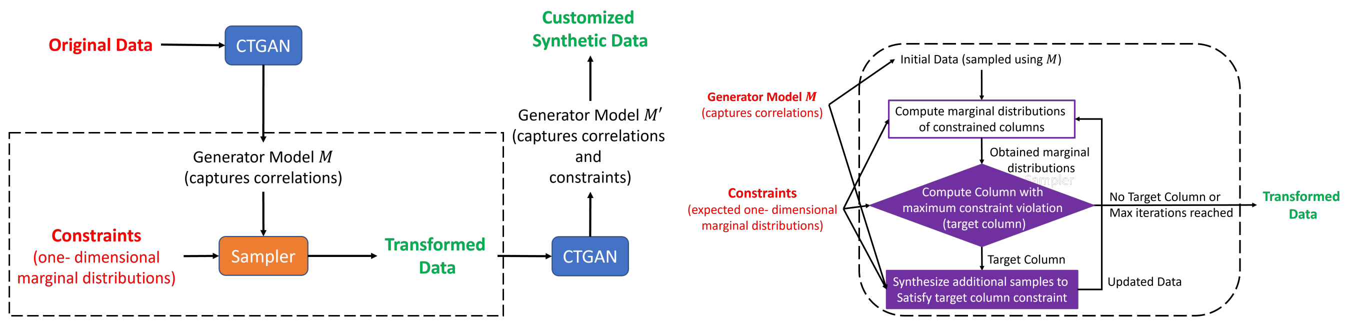
We present CustomGAN, a framework specifically designed to tackle the aforementioned limitations. Our focus lies on classification models applied to tabular data. To the best of our knowledge, there are no existing techniques that address this particular domain. Our framework introduces an algorithm that generates realistic test data while offering a high level of customization. In this paper, we propose a novel solution that effectively captures correlations from the input data and allows for external customizations.

2 RELATED WORK

Data synthesis for tabular data has been extensively studied, leading to the development of various techniques.

Statistical Models involve modeling input data with a multivariate distribution, utilizing techniques like copulas [12, 17], Bayesian networks [23], Gibbs sampling [16], and Fourier decompositions [2] to capture variable dependencies. In contrast, synopses-based approaches like wavelets and multi-dimensional sketches [6] create concise data summaries, facilitating efficient joint distribution estimation and generation.

Neural Models based techniques leverage deep generative models to approximate the input data. Popular techniques in this category include autoencoders [8], variational autoencoders (VAE) [20], and more recently, generative adversarial networks (GANs) [1, 4, 5, 13, 15, 21]. These models enable the generation of synthetic data that captures the underlying distribution patterns of the original data. For instance, CTGAN [22] effectively addresses these challenges with mode-specific normalization, ensuring that the generated data accurately reflects the distribution characteristics of each mode. Furthermore, CTGAN incorporates a conditional generator and employs training-by-sampling techniques to enhance the fidelity of synthetic tabular data. Nevertheless, it's important to note that these techniques lack customization options. There


Original Data
CTGAN
Generator Model M (captures correlations)
Constraints (one- dimensional marginal distributions)
Sampler
Transformed Data
CTGAN
Generator Model M' (captures correlations and constraints)
Customized Synthetic Data
Generator Model M (captures correlations)
Constraints (expected one- dimensional marginal distributions)
Initial Data (sampled using M)
Compute marginal distributions of constrained columns
Obtained marginal distributions
Compute Column with maximum constraint violation (target column)
Target Column
Synthesize additional samples to Satisfy target column constraint
Updated Data
No Target Column or Max iterations reached
Transformed Data

(a) Synthesizer. **(b) Sampler.**

Figure 1: Custom GAN Architecture.

are techniques harnessing the latest transformer technology, eg. REaLTabFormer [19] which incorporates methodologies such as the sequence-to-sequence (Seq2Seq) model and statistical bootstrapping to effectively detect overfitting, setting new standards in synthetic data generation. In contrast, while sophisticated methods like CTGAN [22] and Variational Auto- encoder (VAE) [10] excel in creating realistic synthetic data, they often fall short in providing user-customizable constraints and interpretability to visualize data distributions. Meanwhile, SMOTE [3], a widely used technique, is limited to generating data points within the proximity of existing ones. This highlights the need for a solution that bridges the gap between realism and customization in synthetic data generation.

AITEST [18] introduced an automated framework for testing black-box classification models using tabular data with numeric and categorical columns. While it generates realistic synthetic data and allows user-controlled constraint manipulation, it struggles to capture general dataset dependencies. Complex relationships unaccounted for in the associations limit its applicability across datasets where true functions significantly differ from the constraints. The framework's inability to capture complex column relationships beyond pairwise associations, despite prioritizing categorical columns, hinders its capacity to generate realistic data.

3 CUSTOM TABULAR GAN SYNTHESIZER

In this section, we discuss our overall solution approach spread across the subsequent subsections.

3.1 Architecture

The proposed approach involves multiple steps to generate synthetic data that captures inter-column value-based correlations and adheres to additional constraints. Figure 1 shows our Custom GAN Synthesizer Architecture.

Firstly, a CTGAN model is trained using the original dataset to learn and capture the inter-column correlations. This step ensures that the model can generate synthetic data that reflects the underlying relationships between columns. Next, a sampler is applied which leverages the CTGAN model to produce data and transform it to meet the additional constraints. These constraints are provided as expected marginal distributions for specific columns. By applying the sampler, the data from CTGAN is modified to align with the desired constraints while preserving the inter-column correlations.

Once the data is produced according to the constraints, another CTGAN model is employed. This second CTGAN model operates on the transformed data to further refine the generation process. It allows to generate a dataset for any data-scale while ensuring that the inter-column correlations learned from the original dataset are preserved and the input constraints are also satisfied.

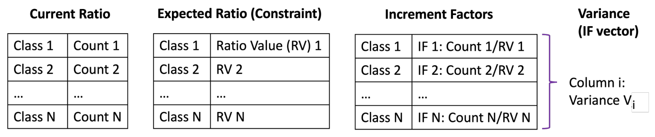
By incorporating these steps, the proposed framework enables the generation of realistic synthetic data that captures inter-column value-based correlations while adhering to user-defined constraints on the marginal distributions of individual columns.

3.2 Sampler

The sampler plays a crucial role in constructing a dataset that effectively captures both the correlations present in the original data and the input constraints, which are defined as one-dimensional marginal distributions. It takes as input a generative model trained on the original dataset and the specified constraints as shown in Figure 1. It analyzes the inter-column correlations present in the original data and utilizes this information to guide the sampling process. By considering the desired one-dimensional marginal distributions, the sampler ensures that the generated dataset adheres to the given constraints.

The Sampler module initiates by generating initial rows from the first CTGAN model trained on the original dataset. It employs an iterative process, as illustrated in Figure 1, to progressively add more rows to the generated data. In each iteration, it identifies the "target column," the one most deviating from the desired marginal distribution in the data generated so far. Then, the sampler uses the CTGAN Model to augment the data with additional rows to align with the desired marginal distribution of the target column. Therefore, as the sampler keeps running, typically the columns tend to become closer to their desired marginal distributions.

Target Column Computation: We now discuss how a target column is computed. To identify the target column, for each of the discrete columns, we compute the number of rows present for each column value. Based on this information and the desired marginal distribution, we compute *Increment Factor* (IF) for each of the column values. This is calculated as $\frac{row-count}{desired-ratio-value}$. The



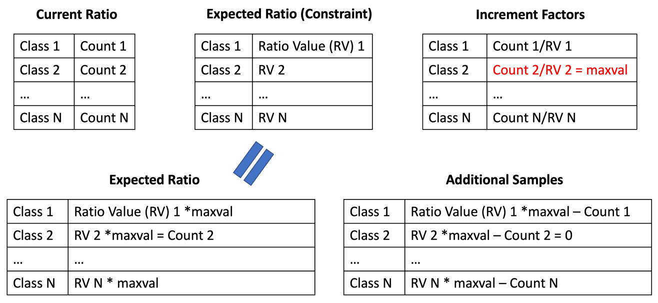
Current Ratio	
Class 1	Count 1
Class 2	Count 2
…	…
Class N	Count N

Expected Ratio (Constraint)	
Class 1	Ratio Value (RV) 1
Class 2	RV 2
…	…
Class N	RV N

Increment Factors	
Class 1	IF 1: Count 1/RV 1
Class 2	IF 2: Count 2/RV 2
…	…
Class N	IF N: Count N/RV N

Figure 2: Target Column Identification

collection of IF values for the values present in a column makes an IF vector for that column. The column for which the variance in IF vector is the most is considered as the target column. This process is also shown in Figure 2.



Current Ratio	
Class 1	Count 1
Class 2	Count 2
…	…
Class N	Count N

Expected Ratio (Constraint)	
Class 1	Ratio Value (RV) 1
Class 2	RV 2
…	…
Class N	RV N

Increment Factors	
Class 1	Count 1/RV 1
Class 2	Count 2/RV 2 = maxval
…	…
Class N	Count N/RV N

Expected Ratio	
Class 1	Ratio Value (RV) 1 *maxval
Class 2	RV 2 *maxval = Count 2
…	…
Class N	RV N * maxval

Additional Samples	
Class 1	Ratio Value (RV) 1 *maxval – Count 1
Class 2	RV 2 *maxval – Count 2 = 0
…	…
Class N	RV N * maxval – Count N

Figure 3: Satisfy Target Column (Discrete) Constraint.

Target Column Constraint Satisfaction: Once the target column is identified, we next add more rows to the data in order to satisfy the constraint on it. To do this, the column value for which the IF value is highest is picked and we multiply the expected ratios for each discrete value with this IF value. Note that since the same constant is multiplied to all the ratio values, this does not impact the ratio values (relatively). Now, this multiplication gives us the number of rows for each discrete value after the additional rows are added. Using this, and the dataset generated thus far, we compute the number of additional rows per discrete value that needs to be generated. This complete process is also illustrated in Figure 3. Since CTGAN is a conditional GAN, generating rows that have a specific discrete column value is straightforward.

To handle discrete columns, the mentioned strategy is effective. For extending support to marginal distribution constraints on continuous columns, an extra pre-processing step, like binning, will be needed. Incorporating this extension is part of our future work.

4 EXPERIMENTAL EVALUATION

We evaluate the performance of our approach on three grounds: (a) *Constraints Satisfaction*, (b) *Coverage*, and (c) *Downstream Model's Prediction Accuracy*.

Work Experience	Constraint	Original	CustomGAN
FALSE	0.875	0.656	0.84
TRUE	0.125	0.344	0.16
Degree Type	Constraint	Original	CustomGAN
Science	0.25	0.274	0.25
Commerce	0.17	0.674	0.166
Others	0.58	0.051	0.583

Table 1: Accuracy on a 2D-Constraint.

	1D	2D	3D	≥4D
Adult	1	4.7	5	5
Student Placements	1	3	4.6	5

Table 2: Average Number of Iterations

Benchmarks and Configuration: We evaluate our approach on open-source data sets Adult [11] and Student placements dataset [7]. Each constraint is associated with a degree denoting the number of columns involved in that constraint. For example, Table 1 shows a 2D-constraint where the columns *work experience* and *degree type* are involved in the constraint. The constraint says that the values `FALSE` and `TRUE` for the column *work experience* occur with ratio $0.875 : 0.125$ and the values `Science`, `Commerce` and `Others` for column *degree type* appear with ratio $0.25 : 0.17 : 0.58$.

The constraints were designed for the two datasets in the following way:

- **Adult.** Six categorical columns were considered while constructing the constraints – Education, Marital-Status, Relationship, Race, Sex, and Income Label. A total of 19 constraints were designed by random ratio generation, comprising five 1D constraints, six 2D constraints, and four constraints each for 3D and 4D.
- **Student Placements.** Here also six categorical columns were considered – High School Specialization, Work Experience, Gender, MBA Specialization, Degree Type, Placed Label. A total of 56 constraints were designed with random ratio generation, comprising six 1D, twelve 2D, seventeen 3D, twelve 4D, five 5D, and four 6D constraints.

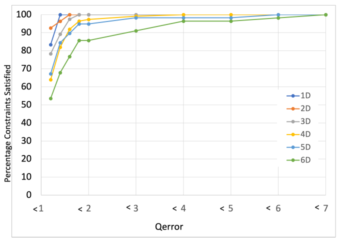
Our code is written in Python and executed in Python 3.10. All the experiments are performed in a machine running macOS 13.4, with 32GB RAM, 10 cores running Apple M1 Pro. For all the constraints across both datasets, we began with an initial dataset of 50 rows before the Sampler iteratively added more rows to satisfy the constraints. We set the limit on the maximum number of allowed iterations to 5. The threshold value for variance, which acts as another termination condition, was set to 1. The first CTGAN model ran for 500 epochs, and the second CTGAN model, which operated after the Sampler produced the transformed data, ran for 100 epochs.

While generating data for one column influences others, potentially causing deviations from marginal distributions, our empirical assessments demonstrate high accuracy within a conservative limit of 5 iterations. Table 2 displays iteration details for various constraints, where 5 iterations represent the limit set by the input parameter, while others result from reaching the error threshold.

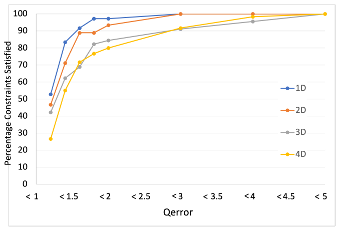
4.1 Constraints Satisfaction

We assess constraint satisfaction accuracy by comparing desired ratios with those in the original and synthetic data (Table 1). Notably, our proposed strategy achieves ratios close to the desired values, as depicted in the figure.

To measure the accuracy, we compute Q-error [14]. For each column value v, its Q-error is calculated as $max(\frac{expected-ratio}{obtained-ratio}, \frac{obtained-ratio}{expected-ratio})$. This quantifies the multiplicative distance between the desired and synthetic ratio values (Figure 4 for student placements and adult dataset). The accuracy for different degree constraints is displayed separately,



(a) Student Placements



(b) Adult

Figure 4: Accuracy on Adult and Student Placements Dataset.

revealing that over 80% of constraints were satisfied with less than 2 Q-error. Additionally, accuracy tends to inversely correlate with constraint degree, logically reflecting increased complexity with higher degrees.

A similar behaviour was observed on comparing the distributions using KL divergence as well. For each constraint, we calculated the KL divergence of the data distribution obtained wrt the expected distribution from the constraint. The results for the two datasets across various constraints are shown in Table 3 and Table 4.

	mean	median	75th percentile	90th percentile	max
1D	0.00	0.00	0.008	0.01	0.01
2D	0.01	0.00	0.01	0.02	0.03
3D	0.02	0.00	0.02	0.04	0.12
4D	0.03	0.01	0.04	0.11	0.20
5D	0.04	0.01	0.04	0.07	0.24
6D	0.09	0.02	0.12	0.22	0.63

Table 3: KL Divergence for Student Placement

	mean	median	75th percentile	90th percentile	max
1D	0.03	0.02	0.042	0.05	0.05
2D	0.04	0.04	0.05	0.07	0.07
3D	0.09	0.03	0.21	0.39	0.60
4D	0.15	0.08	0.17	0.34	0.64

Table 4: KL Divergence for Adult

4.2 Coverage

To perform this experiment, we train a Decision Tree Classifier using the adult dataset and compute the number of paths traversed by the input data. We record the total count of unique decision paths present in this model. We use Decision Tree since it gives coverage of the dataset. We then generate synthetic data with same number of rows as input data and varying number of constraints using CustomGAN and *AITEST*[18]. To the best of our knowledge, AITEST is the only technique where user given customized distributions are preserved, other data synthesis like GANs and VAEs preserve the general data distribution but do not cater the user defined or constrained distribution. The number of constraints are varied from 1D to 4D. We count the number of paths traversed by the data generated by CustomGAN and *AITEST* generated data in the model. Table 5 shows the coverage results for different number of constraints for the *adult* dataset. The numbers shown are average number of paths for data generated with 10 different constraints for each number of constraints. For example, for 10 1D-constraints, i.e. 1 column constraints, the average number of paths traversed by 10 datasets generated using *AITEST* and CustomGAN are 1682 and 1728 respectively. The results clearly shows the number of paths traversed by CustomGAN generated data is much larger than the number of paths traversed by *AITEST*, showing higher coverage.

Constraints	Original	AITEST	CustomGAN
1D	3023	1682	1728
2D	3025	1753	1855
3D	3017	1739	1864
4D	3023	1561	1870

Table 5: Comparison of coverage between Original, AITEST, and CustomGAN

4.3 Prediction Accuracy

To perform this experiment, we split the input data in 80:20 to get *train* and *test*, respectively. We do not use a separate validation set to maximize the size of the training set, particularly when dealing with limited data. This approach helps ensure that the model learns from as much data as possible while still allowing us to assess its performance on unseen data through the test set. We then train a model (Decision Tree, Logistic Regression, Random Forest and Multi Layer Perceptron (MLP)) using the CustomGAN and *AITEST* with different number of constraints, and test the accuracy using *test* data. For all the models (Decision Tree, Random Forest, Logistic Regression, Multi Layer Perceptron), we use default parameters from *scikit-learn*. For example, we use default Decision Tree params from *scikit-learn*, i.e., max_depth=None (the tree nodes continue to expand until all leaves are either pure or contain fewer than the specified "min_samples_split" samples.) , min_samples_split =2 (min. no. of samples needed to split an internal node), min_samples_leaf=2 (min. no. of samples needed to be present in a leaf node). Table 6 shows the results for accuracy results, showing accuracy for the models generated on CustomGAN and *AITEST* generated models. The results clearly shows the accuracy of models trained on customized synthetic data using CustomGAN doing better than the model trained on *AITEST* generated customized data.

These two experiments show CustomGAN not only preserves the given constraints but also preserves the coverage and inherent distribution of the given data.

Con.	Decision Tree			Random Forest			Logistic Regression			MLP		
	Org	AIT	CG	Org	AIT	CG	Org	AIT	CG	Org	AIT	CG
1D	80.8	61.8	67	84	61.5	72.4	79.5	70.8	67.8	77.7	76.5	72.2
2D	80.8	65.4	69.2	84	71.2	75.6	79.5	78.6	73.8	78	76.6	73.5
3D	80.8	62.2	69.5	84.2	67.8	75.8	79.5	74.6	75.7	78.39	51.5	73.8
4D	80.7	60.3	69.3	84.3	71.3	72.3	79.5	76.6	77.4	77.6	52	77.4

Table 6: Comparison of prediction accuracy between Original, AITEST, and CustomGAN

5 CONCLUSION

We present CustomGAN, a framework utilizing large GAN-based synthesis for customizable and realistic synthetic data, balancing constraint satisfaction with data distribution preservation. Our technique has been validated through experimentation, addressing a gap in tabular data classification. Future work includes incorporating explanations to enhance user understanding of the synthetic data's patterns and constraints, along with exploring the utilization of large models in the customized data synthesis process.

REFERENCES

[1] Mrinal Kanti Baowaly, Chia-Ching Lin, Chao-Lin Liu, and Kuan-Ta Chen. 2019. Synthesizing electronic health records using improved generative adversarial networks. *Journal of the American Medical Informatics Association* 26, 3 (2019), 228–241.

[2] Boaz Barak, Kamalika Chaudhuri, Cynthia Dwork, Satyen Kale, Frank McSherry, and Kunal Talwar. 2007. Privacy, accuracy, and consistency too: a holistic solution to contingency table release. In *Proceedings of the twenty-sixth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. 273–282.

[3] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. 2002. SMOTE: synthetic minority over-sampling technique. *JAIR* 16 (2002), 321–357.

[4] Haipeng Chen, Sushil Jajodia, Jing Liu, Noseong Park, Vadim Sokolov, and VS Subrahmanian. 2019. FakeTables: Using GANs to Generate Functional Dependency Preserving Tables with Bounded Real Data.. In *IJCAI*. 2074–2080.

[5] Edward Choi, Siddharth Biswal, Bradley Malin, Jon Duke, Walter F Stewart, and Jimeng Sun. 2017. Generating multi-label discrete patient records using generative adversarial networks. In *Machine learning for healthcare conference*. PMLR, 286–305.

[6] Graham Cormode, Minos Garofalakis, Peter J Haas, Chris Jermaine, et al. 2011. Synopses for massive data: Samples, histograms, wavelets, sketches. *Foundations and Trends® in Databases* 4, 1–3 (2011), 1–294.

[7] Dhimant Ganatara. [n. d.]. Campus Recruitment Analysis. ([n. d.]). https://www.kaggle.com/datasets/benroshan/factors-affecting-campus-placement

[8] Lovedeep Gondara and Ke Wang. 2018. Mida: Multiple imputation using denoising autoencoders. In *Advances in Knowledge Discovery and Data Mining: 22nd Pacific-Asia Conference, PAKDD 2018, Melbourne, VIC, Australia, June 3-6, 2018, Proceedings, Part III 22*. Springer, 260–272.

[9] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In *NIPS*. 2672–2680.

[10] Diederik P. Kingma and Max Welling. [n. d.]. Auto-Encoding Variational Bayes. In *ICLR 2014*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1312.6114

[11] Ronny Kohavi and Barry Becker. [n. d.]. Census Income Data. ([n. d.]). https://archive.ics.uci.edu/ml/datasets/adult

[12] Haoran Li, Li Xiong, Lifan Zhang, and Xiaoqian Jiang. 2014. DPSynthesizer: differentially private data synthesizer for privacy preserving data sharing. In *Proceedings of the VLDB Endowment International Conference on Very Large Data Bases*, Vol. 7. NIH Public Access, 1677.

[13] Pei-Hsuan Lu, Pang-Chieh Wang, and Chia-Mu Yu. 2019. Empirical evaluation on synthetic data generation with generative adversarial network. In *Proceedings of the 9th International Conference on Web Intelligence, Mining and Semantics*. 1–6.

[14] Guido Moerkotte, Thomas Neumann, and Gabriele Steidl. 2009. Preventing bad plans by bounding the impact of cardinality estimation errors. *Proceedings of the VLDB Endowment* 2, 1 (2009), 982–993.

[15] Noseong Park, Mahmoud Mohammadi, Kshitij Gorde, Sushil Jajodia, Hongkyu Park, and Youngmin Kim. 2018. Data synthesis based on generative adversarial networks. *arXiv preprint arXiv:1806.03384* (2018).

[16] Yubin Park and Joydeep Ghosh. 2014. PeGS: Perturbed Gibbs Samplers that Generate Privacy-Compliant Synthetic Data. *Trans. Data Priv.* 7, 3 (2014), 253–282.

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

[18] Diptikalyan Saha, Aniya Aggarwal, and Sandeep Hans. 2022. Data synthesis for testing black-box machine learning models. In *5th Joint International Conference on Data Science & Management of Data (9th ACM IKDD CODS and 27th COMAD)*. 110–114.

[19] Aivin V Solatorio and Olivier Dupriez. 2023. REaLTabFormer: Generating Realistic Relational and Tabular Data using Transformers. *arXiv preprint arXiv:2302.02041* (2023).

[20] Saravanan Thirumuruganathan, Shohedul Hasan, Nick Koudas, and Gautam Das. 2019. Approximate query processing using deep generative models. *arXiv preprint arXiv:1903.10000* (2019).

[21] Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. 2019. Modeling tabular data using conditional gan. *Advances in neural information processing systems* 32 (2019).

[22] Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. 2019. Modeling tabular data using conditional gan. *Advances in Neural Information Processing Systems* 32 (2019).

[23] Jun Zhang, Graham Cormode, Cecilia M Procopiuc, Divesh Srivastava, and Xiaokui Xiao. 2017. Privbayes: Private data release via bayesian networks. *ACM Transactions on Database Systems (TODS)* 42, 4 (2017), 1–41.