**Abstract**

1. **Introduction**

**2. Literature Review**

In the past few years, Large Language Models (LLMs) have become prominent, and with the release of commercial models like ChatGPT by OpenAI in November 2022 [1], their power became available to anyone with internet access, greatly impacting many aspects of daily life [2].

A common belief behind the success of LLMs is the scaling law of computing, model size, and, perhaps most importantly, the high quality of pre-training data [3]. The biggest LLMs today are often pre-trained on trillions of tokens. For example, GPT‑3 was famously trained on nearly 500 billion tokens [4], from a mixture of web text, books, and other sources. GPT‑4 is rumored to have used well over a trillion tokens [5], and Anthropic’s Claude reportedly relies on a similarly large-scale corpus, likely in the hundreds of billions to trillions of tokens [6]. The exact numbers for GPT‑4 and Claude have not been officially disclosed by OpenAI and Anthropic, but external analyses report similar figures [7].

However, acquiring such a massive quantity of high-quality data has become more challenging [8]. Many sources are now gated behind paywalls, restricted by copyright, or filtered due to data quality concerns [9]. As the demand for high-quality training data grows, finding scalable solutions for future LLM development remains an open question.

As a remedy, synthetic data has been widely adopted in training LLMs, offering a more accessible and controllable alternative to real-world data [10, 11]. Chen et al. [3] conducted a study on the measurement of diversity in synthetic data and its impact on LLM performance. They examined how synthetic data diversity influences both pre-training and fine-tuning stages, introducing a new diversity metric called LLM Cluster-Agent, designed specifically to evaluate the diversity of synthetic datasets. They define LLM Cluster-Agent as “a diversity measure pipeline that leverages LLM’s ability to interpret semantic meanings and understand rich contexts of text samples for clustering”. This metric is particularly suited for text-based synthetic data, which is commonly used in the pre-training process of large LLMs, rather than for tabular data. Through a series of controlled experiments with 350M and 1.4B parameter models, Chen et al. demonstrated that higher diversity in synthetic data correlates positively with both pre-training and fine-tuning performance. Interestingly, their findings suggest that synthetic data diversity in pre-training has an even stronger effect on fine-tuning than on pre-training itself. Although this study differs from our goal of comparing tabular data generators rather than synthetic text data, it is still relevant because it highlights how synthetic data can be effectively leveraged in the pre-training of LLMs.

The use of synthetic data generators for training LLMs, however, is not their only application. In fact, synthetic data generation is now widely used across multiple domains. Lu et al. [12] presented a comprehensive review of existing studies on employing machine learning for synthetic data generation, highlighting applications spanning computer vision, speech, natural language processing, healthcare, and business domains. Their review categorizes existing approaches based on machine learning techniques, with a particular emphasis on deep generative models, including GANs, VAEs, and reinforcement learning-based methods. One of the key findings of their study is that the effectiveness of synthetic data depends on the application domain. In computer vision, synthetic datasets are frequently used to train models for object detection, facial recognition, and domain adaptation when real-world labeled data is scarce. In speech processing, synthetic data has proven valuable in speech synthesis and voice cloning, reducing the need for extensive manually labeled datasets. In natural language processing, it is used to augment training datasets for tasks such as language modeling and machine translation. In healthcare, synthetic data generation enables the use of privacy-preserving patient data, facilitating medical research and predictive modeling without compromising sensitive information. In business and finance, synthetic data is used to simulate market behaviors, detect fraudulent transactions, and improve risk assessment models. Beyond applications, Lu et al. also discuss key challenges in synthetic data generation, particularly issues of data fidelity and bias. They emphasize that while synthetic data can approximate real-world distributions, its utility depends on the balance between realism and generalization. Their study provides an important foundation for understanding the broad applicability of synthetic data generation, reinforcing its relevance across various fields where data limitations exist.

Since this study compares existing open-source Python packages for synthetic data generation, it is essential to review the technical aspects and identify the most suitable and widely used models. Various approaches exist for generating synthetic data, ranging from graph-based models and probabilistic methods to deep neural networks. To highlight some of the most well-known models, I refer to the work of Bauer et al. [13], which provides a comprehensive overview of synthetic data generation techniques. This section will not delve into the technical details of each model, as a more precise definition of the models used in the comparison will be presented in the Method section.

Starting with probabilistic and statistical models, one of the most widely implemented is the Gaussian Mixture Model (GMM). GMMs are density estimation algorithms primarily used for clustering, but they can also serve as generative probabilistic models. They are commonly applied to tabular data and time-series generation. A GMM consists of N Gaussian distributions, each representing a continuous, symmetric probability distribution. Another important probabilistic model is the Markov Chain, which is used for generating sequential data by modeling infinite sequences of symbols where the probability of each symbol depends only on the previous n symbols. These models are widely applied in text generation and time-series synthesis.

Bayesian Networks (BNs) offer a graphical approach to modeling dependencies between variables. They are structured as Directed Acyclic Graphs (DAGs), where nodes represent random variables, and edges define their conditional dependencies. Each variable follows either a continuous or discrete probability distribution. Synthcity, one of the Python packages we will analyze, implements Bayesian Networks as they are particularly effective for structured synthetic data generation, including privacy-preserving applications. The second Python package in our study, SDV, utilizes another probabilistic model, the Gaussian Copula. A copula function represents the joint probability distribution of a continuous random vector by separating the individual marginal distributions from the dependency structure between variables. More details on Bayesian Networks and Gaussian Copulas will be provided in the Method section.

Even though probabilistic and statistical models are still widely used, deep learning methods have become the dominant approach for state-of-the-art synthetic data generation. Among them, one of the most well-known frameworks is Generative Adversarial Networks (GANs). GANs consist of two neural networks, a generator (G) that creates synthetic data from random noise and a discriminator (D) that determines whether a given sample comes from the generator or the real training data. The authors of the original GAN paper describe this system as a “minimax two-player game”, where the generator continuously improves its ability to fool the discriminator, while the discriminator becomes better at distinguishing real from fake data [14]. Over time, numerous variations of the classic GAN architecture, originally implemented with Multi-Layer Perceptrons (MLPs), have emerged to improve stability, control, and performance. Deep Convolutional GANs (DCGANs) [15] introduced the use of convolutional layers instead of fully connected layers, allowing the generator to better capture spatial hierarchies in data, significantly enhancing the quality of image generation. Conditional GANs (cGANs) [16] addressed the uncontrolled nature of GAN outputs by introducing conditioning variables, such as class labels or additional attributes, enabling the generator to produce targeted synthetic samples. Another major advancement came with Wasserstein GANs (WGANs) [17], which improved training stability by replacing the traditional Jensen-Shannon divergence with the Wasserstein distance, mitigating common issues such as mode collapse and leading to more reliable convergence.

Another widely used deep learning-based approach for synthetic data generation is Variational Autoencoders (VAEs). VAEs are probabilistic generative models designed for latent space learning, enabling the generation of high-dimensional synthetic data such as images and text. Unlike GANs, which learn to generate data through adversarial training, VAEs model the data distribution explicitly by encoding inputs into a latent space and then reconstructing them via a decoder [18]. While VAEs do not always produce sharper images compared to GANs, they offer greater control over latent variables, making them useful for tasks requiring structured and interpretable representations.

For image and text synthesis, powerful generative models are Diffusion Models [19]. These models operate as Markov chains, where data is incrementally noised in a forward process over T steps, and the model learns to reverse this process, gradually denoising the input back to the original data distribution. Diffusion models have gained attention for their ability to generate highly detailed images, surpassing GANs in certain text-to-image tasks.

Perhaps the most influential deep learning model in text synthesis, and beyond, is the Transformer architecture. First introduced in 2017 in the seminal paper “Attention Is All You Need” [20], Transformers gained widespread recognition following the release of LLMs such as ChatGPT, which are built upon Transformer-based architectures. At their core, Transformers are sequence-to-sequence transduction models structured with an encoder-decoder mechanism. Unlike previous recurrent architectures (RNNs and LSTMs), Transformers allow for full parallelization, drastically improving efficiency and scalability. The key innovation behind Transformers is the multi-head self-attention mechanism, which enables models to capture long-range dependencies in data with a constant number of sequential operations, rather than the sequential processing bottleneck of RNNs. This shift allowed Transformers to excel in language modeling, translation, and generative tasks, building the foundation for modern LLMs.

To narrow the scope and examine studies similar to this one, as the conclusion of this literature review, we will analyze research comparing synthetic data generation techniques in real-world applications. One such study is conducted by Akiya et al. [21], which evaluates various synthetic data generation methods for control group survival data in oncology clinical trials. The primary objective of their research was to determine the most suitable synthetic patient data (SPD) generation method for oncology trials, focusing specifically on progression-free survival (PFS) and overall survival (OS), key evaluation endpoints in clinical oncology. In their study, Akiya et al. compared four distinct synthetic data generation techniques, incorporating both probabilistic/statistical methods and deep learning-based approaches. The traditional methods included Classification and Regression Trees (CARTs) and Random Forest (RF), while more complex models consisted of Bayesian Networks (BNs) and Conditional Tabular Generative Adversarial Networks (CTGANs). To evaluate performance, the researchers generated 1,000 synthetic datasets per method and assessed their effectiveness based on both statistical similarity and visual analysis. The results indicated that traditional tree-based methods outperformed deep learning-based techniques, particularly when trained on relatively small datasets, which is common in clinical trials. CART and RF demonstrated superior performance, with CART emerging as the most effective method, as its synthetic data closely matched the statistical properties of real patient survival data. On the other hand, Bayesian Networks (BNs) and CTGANs did not perform well, mainly due to their higher data requirements. These models typically require larger training datasets to learn meaningful patterns and generate synthetic data that aligns well with real-world statistical distributions.

While the previous study provides insights into synthetic data generation techniques, comparisons between specific open-source Python packages remain scarce. This study aims to fill that gap in the literature by providing a comparative analysis of two of the most popular open-source Python libraries for synthetic data generation: SDV and Synthcity. To evaluate these two packages, we will compare different models available in each framework. For SDV, we will analyze the Gaussian Copula synthesizer, Conditional Tabular GAN (CTGAN), and Tabular Variational Autoencoder (TVAE). For Synthcity, we will evaluate the Bayesian Network synthesizer, as well as CTGAN and TVAE, to ensure a direct comparison between the shared models across both packages. The synthetic data will be generated using a dataset from a publicly available repository in the UCI Machine Learning repository. It was collected by continuously monitoring a low-energy house in Belgium for 137 days, capturing both electrical energy consumption and environmental data. To assess the quality and effectiveness of the generated data, we will employ two key evaluation metrics:

1. Statistical Difference: Measured by comparing synthetic and real data distributions using reliable statistical functions provided by SDV, along with custom statistical comparison methods.
2. Predictive Utility: Evaluated by training Machine Learning models on both real and synthetic data and comparing the performance metrics (e.g., accuracy, precision, recall) of the models trained on each dataset.

This study aims to identify the best-performing open-source package for synthetic data generation and provide a comprehensive comparison of the quality and utility of the data generated by SDV and Synthcity. By doing so, we hope to contribute valuable insights into the strengths and limitations of these tools, guiding researchers and practitioners in selecting the most suitable synthetic data generation framework for their needs.

1. **Methods**
   1. **Data Generators Description**

In this research, we'll use four different Data Generators with well-known mathematical properties: Gaussian Copula, Bayesian Networks, Conditional Tabular GAN (CTGAN), and Tabular Variational Autoencoder (TVAE). The first two are statistical methods, whereas the latter two are deep neural network techniques. Let's first examine the mathematical properties of the statistical methods.

Gaussian copula methods model the joint distribution of a table by combining each column’s marginal distribution with a copula function capturing inter-column dependencies. The foundation is Sklar’s theorem [22], which states any multivariate distribution can be decomposed as

where are the marginals and is the copula describing their dependency structure. A Gaussian copula assumes this dependence is encoded by a multivariate normal distribution with a correlation matrix, while allowing arbitrary marginals. In practice, one first estimates each column’s CDF, then transforms the data into a latent space with uniform or Gaussian marginals. First, you estimate a CDF for each column of the real data. Then, transform the real data into uniform variables and fit a copula to capture their joint dependence, which for a Gaussian copula means computing the correlation matrix. Lastly, draw a synthetic sample from the copula model, and invert the transform by applying the inverse CDF for each column, This yields a synthetic data record in the original space.

Patki et al. [23] introduced the open-source package Synthetic Data Vault (SDV) by using Gaussian copulas to model tabular data. In their approach, all columns are converted to a standard normal scale to remove the effect of each column distribution shape, before estimating the covariance matrix of the joint Gaussian copula. After modeling the correlations, synthetic rows are sampled by drawing from the multivariate normal and then transforming back to each column’s domain. One limitation, however, is that purely categorical fields cannot be directly handled by the Gaussian copula, since the copula operates in a continuous space. The SDV work addressed this by encoding categories as ordinal values in the range [0,1] so that they could be treated like continuous variables in the copula model. In general, the key advantage of Gaussian copula models is that they are statistical methods, as opposed to neural networks, offering a relatively simple mathematical formulation and a stable fitting procedure, while still capturing complex dependencies through the copula. However, if the data contains many discrete variables or highly non-linear dependencies, a copula model, like the Gaussian copula, might struggle.

Another class of tabular synthesizers uses Bayesian networks (BN) to learn the joint distribution from the data. A Bayesian network consists of a directed acyclic graph, in which the nodes represent variables of the data (columns of a data frame), and a set of conditional probability distributions for each node given its parent nodes [24]. In essence, the BN factorizes the joint probability

into ,

Representing inter-column relationships as conditional dependencies. Once a BN is learned from the real dataset, generating synthetic data is simple: one can sample from the network by first sampling the root nodes, which are variables with no parents according to the learned graph, and then sampling descendant nodes conditional on their parents’ sampled values, propagating through the network until all variables have values. This produces a synthetic record that statistically mirrors the correlation captured in the BN.

An example of Bayesian Network usage can be found in the research by Zhang et al. [25], which introduces PrivBayes, a method utilizing Bayesian networks to generate synthetic data under differential privacy constraints. PrivBayes learns a dependency graph over the attributes (columns) and then draws synthetic tuples (rows) by sampling that Bayesian Network. In general, a learned BN can accurately reproduce multi-variable interactions present in the original data, especially for mixed categorical data. Therefore, BN have the advantage of being well-understood models in probability theory, and they inherently ensure that generated samples are consistent with the conditional distributions observed in the real data. On the other hand, Bayesian network generators face some practical challenges. In fact, they often require discretizing continuous variables or assuming parametric forms for continuous conditional distributions, which can introduce errors [25]. Moreover, learning the optimal network structure for high-dimensional data can be computationally expansive and may require prior knowledge.

On the other hand, compared to statistical models such as the Gaussian Copula and Bayesian networks, deep learning methods exist for generating tabular synthetic data. As previously stated, this study will utilize the Conditional Tabular GAN (CTGAN) and Tabular Variational Autoencoder (TVAE). CTGAN is a deep generative model specifically designed for tabular data, introduced by Lei Xu and colleagues in 2019 [26]. It extends the standard Generative Adversarial Network architecture, consisting of a generator that synthesizes data and a discriminator that attempts to distinguish between real and synthetic data. CTGAN introduces novel components to tackle particular challenges of tabular data, such as mixed data types, imbalanced categories, and complex distributions. Unlike GAN architectures commonly used in image generation, which typically rely on convolutional networks [27], CTGAN employs fully connected multilayer perceptrons for both its generator and discriminator.

Lei Xu et al. introduce three key technical innovations in the CTGAN framework: Mode-Specific Normalization, a Conditional Generator, and a Training-by-Sampling strategy. Mode-Specific Normalization refers to CTGAN’s use of variational Gaussian Mixture models to preprocess continuous columns. Rather than employing simple min-max scaling, as often seen in classic GAN implementations [28], each continuous column is modeled as a mixture of Gaussians. Each value is normalized according to the “mode” (Gaussian component) it most likely belongs to. This normalization strategy helps the generator learn multi-modal and non-Gaussian distributions by providing a richer and more expressive representation of the continuous data. Essentially, it maps a continuous variable into a higher-dimensional space (with dimensions corresponding to mixture components), effectively addressing issues related to non-Gaussian distributions.

Another significant innovation of CTGAN is the Conditional Generator. Unlike traditional methods that generate entire rows unconditionally, CTGAN conditions the generation process on specific discrete column values during training. For example, if a table has a categorical column labeled "Education" with categories such as "High School," "Bachelor," and "Master," the model may fix a specific category (e.g., "Master") during certain training iterations and train the generator to produce rows conditioned on this category. This conditioning is implemented by appending a one-hot vector to the generator's input to indicate the selected category and by filtering real data accordingly when updating the discriminator. Consequently, the GAN explicitly learns the conditional distributions for each category within a categorical column, significantly improving the fidelity of synthetic data for imbalanced categories.

Accompanying the conditional generator, CTGAN incorporates a Training-by-Sampling strategy to select training minibatches in a balanced manner. When conditioning on a particular discrete category, the method selects real data batches exclusively from rows containing that category to update the discriminator, rather than randomly sampling from the entire dataset. This targeted sampling approach aligns the generator’s conditioning with the discriminator’s data distribution, preventing the generator from being biased toward majority classes. Thus, the GAN effectively trains on one sub-population at a time, implementing a form of oversampling for rare categories.

Regarding its architecture, CTGAN employs the Wasserstein GAN objective with gradient penalty (WGAN-GP) to ensure stable training [29] and incorporates PacGAN [30], which modifies the discriminator to jointly evaluate multiple samples, thus further mitigating mode collapse. The architecture combines a noise input z with the aforementioned condition vector, feeding the generator to produce synthetic rows. The discriminator then attempts to distinguish between real and synthetic data. Thanks to Mode-Specific Normalization, the generator’s output for each continuous feature can be accurately mapped back to the real data space, preserving the original distributions.

In the same paper in which Xu et al. [26] introduced the CTGAN for the first time, they also presented a the Tabular Variational Autoencoder (TVAE), a deep generative model that applies Variational Autoencoder (VAE) framework to tabular data. TVAE adapts a standard variational autoencoder to handle mixed data types. In a VAE, there are two networks: an encoder that maps an input data sample (a row in the table) to a latent representation , and a decoder that tries to reconstruct the original data from the latent code. The model is trained by maximizing the evidence lower bound (ELBO), which consists of a reconstruction term, ensuring that the decoded output matches the input data, and a regularization term pushing the latent to follow a specific standard distribution, usually standard normal [31].

To make VAEs work for tabular data (TVAE) Xu et al. use similar data pre-processing as for CTGAN, for instance by converting categorical columns into one-hot vectors and using Mode-Specific Normalization for continuous columns, so that the encoder and decoder can effectively model them. The decoder network in TVAE outputs parameters sufficient to reconstruct each column: for continuous columns, it might output means/variances (if assuming a Gaussian output distribution), and for categorical columns, it can output logits for each category, i.e. softmax probabilities for one-hot output. On the other hand, the encoder network likewise must handle one-hot inputs for categorical columns and numeric inputs for continuous columns. With this implementation, TVAE is able to capture the joint distribution of heterogeneous tabular columns in its latent space. Once trained, synthetic data generation is done by sampling a latent vector from the prior (e.g. ) and feeding it into the decoder to produce a new synthetic row. Because the decoder was trained to produce realistic combinations of values, in order to match the real data distribution, the sample outputs resemble real records. Xu et al. also demonstrated that a “vanilla” VAE, when trained with the right loss functions and data encoding, can be competitive with GAN-based models for tabular data. TVAE’s mathematical formulation is essentially the VAE probabilistic model: it seeks to maximize

over the real data, thereby learning which can be sampled. Yadav, Parul, et al. [31] concisely summarize TVAE as “a novel VAE for tabular data using two neural networks (encoder and decoder) trained with ELBO loss”. In practice, TVAE offers a more straightforward training process than GANs, i.e. no adversarial game to balance, and can be easier to converge, though it may require careful tuning to get the decoder to model categorical distributions accurately [31].

* 1. **Comparison process and methods**

The goal of this study is to compare the models and resources offered by two of the most famous Python synthetic data generator packages: SDV and Synthicity. A total of 6 models were chosen to perform the performance comparison, three for SDV and three for Synthicity. Gaussian Copula, CTGAN, and TVAE for SDV, and Bayesian Network, CTGAN, and TVAE for Synthicity, for which a more detailed description is given in the above section. Before to state how we performed the models comparison, it’s important to mention something about the usability and documentation of the two packages.

**References**

[1] OpenAI. *ChatGPT.* 2022. Retrieved from <https://openai.com/index/chatgpt/>.

[2] A. Shaji George, & A. S. Hovan George. (2023). A Review of ChatGPT AI’s Impact on Several Business Sectors. *Partners Universal International Innovation Journal*, *1*(1), 9–23. <https://doi.org/10.5281/zenodo.7644359>

[3] Chen, Hao, et al. "On the Diversity of Synthetic Data and Its Impact on Training Large Language Models." *arXiv* preprint arXiv:2410.15226v2 [cs.CL], 22 Oct. 2024.

[4] Brown, Tom, et al. “Language Models Are Few-Shot Learners.” *Advances in Neural Information Processing Systems*, vol. 33, Curran Associates, Inc., 2020, pp. 1877–1901, <https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967411061b3e5a3861b-Paper.pdf>.

[5] OpenAI. *GPT‑4 Technical Report.* 14 Mar. 2023, <https://cdn.openai.com/papers/GPT-4-System-Card.pdf>.

[6] Bai, Yuntao, et al. *Constitutional AI: Harmlessness from AI Feedback.* Anthropic, 2022, <https://arxiv.org/abs/2212.08073>.

[7] Bubeck, Sébastien, et al. “Sparks of Artificial General Intelligence: Early Experiments with GPT‑4.” *arXiv* preprint arXiv:2303.12712 [cs.LG], 2023.

[8] Pablo Villalobos, Jaime Sevilla, Lennart Heim, Tamay Besiroglu, Marius Hobbhahn, and Anson Ho. Will we run out of data? an analysis of the limits of scaling datasets in machine learning. arXiv. preprint arXiv:2211.04325, 2022.

[9] Perełkiewicz, Michał, and Rafał Poświata. "A Review of the Challenges with Massive Web-mined Corpora Used in Large Language Models Pre-Training." arXiv preprint arXiv:2407.07630 (2024).

[10] Lisa Bauer and Mohit Bansal. Identify, align, and integrate: Matching knowledge graphs to commonsense reasoning tasks. arXiv preprint arXiv:2104.10193, 2021

[11] Ruibo Liu, Jerry Wei, Fangyu Liu, Chenglei Si, Yanzhe Zhang, Jinmeng Rao, Steven Zheng, Daiyi Peng, Diyi Yang, Denny Zhou, et al. Best practices and lessons learned on synthetic data for language models. arXiv preprint arXiv:2404.07503, 2024.

[12] Lu, Yingzhou, et al. *Machine Learning for Synthetic Data Generation: A Review.* arXiv preprint, arXiv:2302.04062, 30 June 2024.

[13] Bauer, André, et al. *Comprehensive Exploration of Synthetic Data Generation: A Survey.* University of Chicago, University of Würzburg, University of Ulm, Argonne National Laboratory, 2024. arXiv preprint arXiv:2401.02524.

[14] Goodfellow, Ian, et al. "Generative adversarial nets." *Advances in Neural Information Processing Systems* (NeurIPS), vol. 27, 2014, pp. 2672–2680.

[15] Radford, Alec, Luke Metz, and Soumith Chintala. "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks." *arXiv preprint* arXiv:1511.06434, 2015.

[16] Mirza, Mehdi, and Simon Osindero. "Conditional generative adversarial nets." *arXiv preprint* arXiv:1411.1784, 2014.

[17] Arjovsky, Martin, Soumith Chintala, and Léon Bottou. "Wasserstein GAN." *arXiv preprint* arXiv:1701.07875, 2017.

[18] Kingma, Diederik P., and Max Welling. "Auto-Encoding Variational Bayes." *arXiv preprint* arXiv:1312.6114, 2013.

[19] Ho, Jonathan, Ajay Jain, and Pieter Abbeel. "Denoising diffusion probabilistic models." *Advances in Neural Information Processing Systems* (NeurIPS), vol. 33, 2020, pp. 6840–6851, arXiv:2006.11239.

[20] Vaswani, Ashish, et al. "Attention Is All You Need." *Advances in Neural Information Processing Systems* (NeurIPS), vol. 30, 2017, pp. 5998–6008.

[21] Akiya, Ippei, Takuma Ishihara, and Keiichi Yamamoto. "Comparison of Synthetic Data Generation Techniques for Control Group Survival Data in Oncology Clinical Trials: Simulation Study." JMIR Medical Informatics, vol. 12, 2024, e55118. <https://doi.org/10.2196/55118>.

[22] Jaworski, P., Durante, F., Hardle, W. K., & Rychlik, T. (2010). Copula theory and its applications (Vol. 198). New York: Springer.

[23] Patki, Neha, et al. "The Synthetic Data Vault." *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, IEEE, 2016. *IEEE Xplore*, doi:10.1109/DSAA.2016.49.

[24] Young, Jim, et al. "Using Bayesian Networks to Create Synthetic Data." *Journal of Official Statistics*, vol. 25, no. 4, 2009, pp. 549–567.

[25] Zhang, Jun, et al. "PrivBayes: Private Data Release via Bayesian Networks." *ACM Transactions on Database Systems*, vol. 42, no. 4, 2017, pp. 1-41.

[26] Xu, Lei, et al. "Modeling Tabular Data Using Conditional GAN." *Advances in Neural Information Processing Systems (NeurIPS 2019)*, Vancouver, Canada, 2019.

[27] Radford, Alec, et al. "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks." *arXiv preprint arXiv:1511.06434* (2015).

[28] Veeramachaneni, Kalyan, et al. "CTAB-GAN+: Enhancing Tabular Data Synthesis." *Frontiers in Big Data*, vol. 6, 2023.​

[29] Gulrajani, Ishaan, et al. "Improved Training of Wasserstein GANs." *Advances in Neural Information Processing Systems*, vol. 30, 2017, pp. 5767–5777.

[30] Lin, Zinan, et al. "PacGAN: The Power of Two Samples in Generative Adversarial Networks." *Advances in Neural Information Processing Systems*, vol. 31, 2018, pp. 14953–14963.

[31] Yadav, Parul, et al. "Rigorous Experimental Analysis of Tabular Data Generated using TVAE and CTGAN." *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 15, no. 4, 2024.