Synthetic Data Generation in Cybersecurity: A Comparative Analysis

Dure Adan Ammara
Department of Computer Science
Blekinge Institute of Technology
Karlskrona, Sweden
dure.adan.ammara@bth.se

Jianguo Ding
Department of Computer Science
Blekinge Institute of Technology
Karlskrona, Sweden
jianguo.ding@bth.se

Kurt Tutschku
Department of Computer Science
Blekinge Institute of Technology
Karlskrona, Sweden
kurt.tutschku@bth.se

Abstract—Synthetic data generation faces significant challenges in accurately replicating real data, particularly with tabular data, where achieving high fidelity and utility is critical. While numerous methods have been developed, the most effective approach for creating high-quality synthetic data for network traffic security remains to be seen. This study conducts a comprehensive comparative analysis of non-AI, conventional AI, and generative AI techniques for synthetic tabular data generation using two widely recognized cybersecurity datasets: NSL-KDD and CICIDS-2017. Particular emphasis was placed on prominent GAN models for tabular data generation, including CTGAN, CopulaGAN, GANBLR++, and CastGAN. The results indicate that GAN-based methods, particularly CTGAN and CopulaGAN, outperform non-AI and conventional AI approaches in terms of fidelity and utility. To the best of our knowledge, this research contributes to the field by offering the first comparative evaluation of these methods specifically for cybersecurity network traffic data, filling a critical gap in the literature. It also introduces mutual information for feature selection, further enhancing the quality of the generated synthetic data. These findings provide valuable guidance for researchers seeking the most suitable synthetic data generation method in cybersecurity applications.

Index Terms—Generative Adversarial Networks (GANs), Synthetic tabular Data, Cybersecurity, NSL-KDD, CICIDS, Mutual information, Feature selection

I. INTRODUCTION

In the rapidly evolving field of cybersecurity, IDS plays a crucial role in identifying and mitigating threats within network environments. These systems rely heavily on high-quality data to detect patterns indicative of malicious activity [1]. However, acquiring real-world cybersecurity data poses significant challenges, primarily due to privacy concerns and time constraints [2]. Researchers have turned to synthetic data generation as a viable alternative to address this. Synthetic data, when adequately generated, can mimic the characteristics of real-world data while avoiding the associated ethical and legal issues, making it a valuable resource for training and evaluating IDS [3].

Despite the growing interest in synthetic data generation, producing synthetic datasets that maintain the utility, fidelity, and data integrity required for practical cybersecurity applications remains a significant challenge [4]. Utility refers to the ability of synthetic data to support the same analytical tasks as real data [5]. At the same time, fidelity ensures that the synthetic data accurately reflects the statistical properties of the original data [6]. Data integrity, on the other hand, focuses on the accuracy and reliability of data, ensuring that it remains unaltered and consistent [7], [8]. Follow these criteria to avoid synthetic data that is either unrealistic or similar to real data, rendering it less valuable or risky.

GANs have shown great potential for generating synthetic tabular data for cybersecurity purposes. Various GAN architectures like VanillaGAN [9], WGAN [10], and WGAN-GP [11], as well as models specifically designed for tabular data generation such as CTGAN [12], CopulaGAN [13], and GANBLR [14], have been successfully used to create synthetic IDS data [15]. However, the effectiveness of GANs for generating tabular data specific to cybersecurity, such as network flow data used in IDS, has yet to be thoroughly benchmarked against other tabular synthetic data generation methods: Artificial intelligence (AI) based and non-AI based.

A. Synthetic data for cybersecurity

1) **Data privacy**: Sharing real-world cybersecurity data involves a risk of sharing sensitive information, which hinders collaboration and research [16]. Thus, the synthetic data resolves this issue, and valuable data can be shared without compromising privacy [17].

2) **Data scarcity**: The real-world cyber security data is often imbalanced, with normal traffic making up a higher percentage than attacks or malicious traffic. This causes a problem of data scarcity. Synthetic data in this scenario can help balance both instances and improve cyber security tasks, such as efficient, unbiased intrusion detection [1].

3) **Diversity of scenarios**: Cybersecurity threats are becoming more dynamic and diverse, but real cybersecurity datasets often need more diversity [18]. In this scenario, synthetic data helps simulate various uncommon attacks and allows for more comprehensive testing of cybersecurity tools [19].

4) **Enhancing AI models**: When training cyber security models such as Intrusion detection systems (IDS), a large, high-quality dataset is essential for effective detection. Therefore, synthetic data plays a crucial role in

training these models on a diverse and extensive dataset to enhance the robustness of AI models [20], [21].

B. Synthetic data challenges in cybersecurity

1) **Data realism**: One of the primary challenges in generating synthetic data is ensuring its realism compared to real-world data [22]. The effectiveness of AI models trained on synthetic data depends on how accurately the data represents real cyber threats and network conditions [23]. Many generative models are still in the process of achieving data realism by increasing their fidelity and utility to mimic actual trends [24].

2) **Lack of variability**: Some generative models are explicitly designed to generate a diverse set of network instances and cyber threats. However, if synthetic data is not generated with sufficient variability, it may miss capturing the full spectrum of possible attack vectors in cybersecurity [25], [26]. For instance, the Wasserstein GAN has solved the problem faced by the vanilla GAN of mode collapse, thus ensuring that the generated outcome is not limited to a few instances [10].

3) **Class imbalance**: Cybersecurity often suffers from class imbalance, where regular traffic is much larger than cyberattacks [27]. Synthetic data generation methods address this problem by balancing malicious and benign traffic and attacks within the cybersecurity dataset for more robust intrusion detection [28].

4) **Privacy concerns**: Synthetic data attempts to conceal confidential information from real-world sensitive data to address privacy and security concerns [29]. However, it is not entirely free from these challenges. Ensuring that sensitive information cannot be traced back from the synthetic data is crucial and remains a concern.

5) **Standardization**: More standardization is needed for generating, validating, and integrating synthetic data into cybersecurity practices. This makes it challenging to ensure the reliability and consistency of the synthetic datasets across different cybersecurity applications [30], [31].

6) **Perfomace evaluation**: For evaluating synthetic data, it is crucial to note that achieving a high similarity score does not guarantee its real-world applicability. Similarly, obtaining high accuracy in synthetic cybersecurity datasets for any ML/DL model does not ensure compatibility with real-world datasets [24], [32].

C. Motivation

This research addresses critical challenges in strengthening IDS defenses amidst a rapidly escalating cyber threat landscape. IDS are vital for securing networks by detecting and mitigating intrusions, but a lack of high-quality, real-world data for training and refinement hampers their effectiveness. Obtaining such data is challenging due to privacy concerns, ethical issues, and evolving network behaviors. Synthetic data generation using techniques like GANs offers a promising solution by creating realistic, high-fidelity data that mimics net-

work activities without violating privacy or legal constraints. This research is crucial to improving IDS capabilities, enabling them to adapt to modern cyber adversaries' sophisticated tactics. This study aims to enhance IDS robustness and reliability by benchmarking different synthetic data generation methods, contributing to more secure cybersecurity frameworks.

This research fills a gap in cybersecurity by offering a comprehensive comparison of synthetic data generation methods for enhancing IDS. While synthetic data's potential has been recognized, there is limited clarity on which methods—traditional, AI-based, or GAN-based—provide the highest fidelity and utility for IDS. Existing literature has explored various techniques but lacks a focused comparison of their effectiveness in generating network traffic data for IDS. This study bridges the gap by systematically evaluating different synthetic data generation techniques, guiding the development of more effective IDS and contributing to more robust, more adaptable cybersecurity defenses.

D. Objectives

The primary objective of this study is to rigorously evaluate and compare the performance of various synthetic data generation methods for cybersecurity, focusing on their utility, fidelity, data integrity, and ability to handle class imbalance. To achieve this, the research addresses the following key questions:

1) Which methods produce synthetic data that best preserves the utility for training IDS models?
2) How well do these methods maintain the original data's statistical properties (fidelity) and integrity?
3) Which techniques effectively balance class distributions in the generated data?
4) Which GAN models are best for producing cyber security network traffic data? These questions guide the comprehensive benchmarking of both statistical and AI-based synthetic data generation techniques.

E. Contribution

The contributions of this work are summarized below:

1) **Comprehensive Comparison of Synthetic Data Generation Methods:** This research uniquely compares three prominent synthetic data generation methods—statistical, classical AI, and generative AI—on cybersecurity network traffic datasets (NSLKDD, CICIDS17). This provides the first direct comparison of these methods in the cybersecurity field.
2) **Filling the Knowledge Gap in Cybersecurity Data:** The study addresses a gap in the literature by identifying the most suitable synthetic data generation method specifically for network traffic datasets, offering clarity in an area where no previous comparative study exists.
3) **Practical Guidance for Researchers:** By evaluating the performance of these methods, this research offers a clear recommendation on the best approach for generating synthetic cybersecurity data, enabling researchers to

make informed decisions without having to test multiple methods.

4) **Introduction of Mutual Information for Feature Selection:** Using Mutual Information for feature selection in synthetic data generation is an innovative contribution, providing a new perspective that may improve feature selection strategies in future cybersecurity studies.

The source code of this experiment is publicly available on our GitHub Repository [1].

F. Paper Organization

The remaining sections of this paper are organized as follows:

- Section II provides an overview of Synthetic Tabular Data Generation Methods, their usage in tabular data applications in cybersecurity, and evaluation metrics.
- Section III details the prominent standard non-AI and AI-based synthetic tabular data generation methods and their mathematical workflow.
- Sections IV and V describe the mutual information and evaluation metrics used in this experiment.
- Section VI discusses the experimental setup, including the datasets and model configurations.
- Section VII presents our experiments' results, highlighting each method's performance across two datasets.
- Finally, in Sections VIII and IX, we mention the potential areas for future work and development, focusing on critical aspects that require further investigation and improvement, and conclude with key highlights of this study.

II. RELATED WORK

A. Synthetic Tabular Data Generation Methods

Synthetic data is an artificially produced dataset that mimics the statistical features of real-world data without disclosing any sensitive information [3]. This data type spans various forms, such as tabular data, images, and text.

Tabular data, organized in a table format with rows and columns, is widely used for ML and DL modeling. It is also crucial for cybersecurity applications due to its structured nature. It is ideal for representing logs, transaction records, and other important information for detecting and mitigating security threats [16]. For example, cybersecurity datasets often contain tabular data in IDS, fraud detection, and anomaly detection, enabling precise analysis and decision-making [33], [34]. Additionally, tabular data's clear and organized format allows for effectively applying ML and DL algorithms to enhance security measures [22].

The significance of synthetic data in cybersecurity stems from the challenges associated with accessing high-quality, real-world data in this field. Privacy concerns, regulatory limits, and the sensitive nature of cybersecurity information often restrict the availability of data [35]. By facilitating the generation of realistic datasets free of genuine, sensitive

information, synthetic data offers a solution to these challenges [36]. This advancement is critical for pushing forward research, enhancing security models, and executing thorough threat analyses.

B. Applications in Cybersecurity

Synthetic data finds multiple applications in cybersecurity, improving different facets of security management and threat response:

1) **Intrusion Detection Systems (IDS):** By supplying varied and representative datasets encompassing a range of attack patterns and behaviors, synthetic data aids in the training and testing of IDS, enhancing their detection accuracy and resilience [37]. Research has shown that employing synthetic data techniques on datasets like UNSW-NB15 and CICIDS-17 has significantly improved the performance of deep learning models used in IDS [38].

2) **Anomaly Detection:** Synthetic data plays a role in simulating typical and atypical behaviors within networks, thereby facilitating the training of systems to detect deviations indicative of potential security incidents [39]. Synthetic data has been used to preserve privacy and detect anomalies in the KDD dataset. Its effectiveness, measured in supervised, semi-supervised, and unsupervised anomaly detection, showed an accuracy of 99.69%, similar to models trained on original data [40].

3) **Security Training and Simulation:** The value of synthetic data extends to crafting realistic scenarios for security training exercises and simulations. This enables security teams to hone their responses to varied cyber threats without compromising real data [41]. A deep Convolutional Generative Adversarial Network (DC-GAN) model was used to generate synthetic cyberattack data and realistic attack simulations to train deep learning models for better detection. The model with those simulated attacks achieved an accuracy of 99.69% on the KDD dataset and 97.93% on the CICIDS-17 dataset [42].

In discussing the significance and uses of synthetic tabular data in cybersecurity, we will focus on the methods used to generate such data effectively. The later sections will delve into the specifics and their architecture. They will also provide an experimental analysis of the prominent standard (non-AI) and AI-based methods for creating synthetic tabular data for cybersecurity IDS.

C. Metrics for Evaluation

1) **Fidelity**: Fidelity measures how well the synthetic data replicates the statistical properties of the original real-world data [43]. Therefore, the single most significant metric for evaluating synthetic tabular data generally involves assessing the data's underlying distribution [44]. This approach is based on the definition of data synthesis to capture the probability distribution function of the

original data, thereby creating synthetic data that mirrors these characteristics [45]. However, the evaluation process can be subjective, based on the data production's objective. As a result, researchers employ various methods to assess synthetic tabular data, considering its structure markedly differs from that of images.

Unlike synthetic tabular data, the validation of synthetic image-based output is often visual. If the model yields realistic images, it's deemed successful. Images comprise spatial information, shape, texture, objects, and color [46]. Nonetheless, synthetic data is confined to images and encompasses tabular data [47], characterized by its complexity, multiple interconnected features, hidden patterns, and context-dependent information [48]. Hence, evaluating synthetic tabular data demands a more structured approach.

2) **Utility**: Utility measures the usefulness of synthetic data for the same tasks or analyses as the original data [49]. To evaluate the quality of the synthetic data, key performance metrics such as accuracy, F1 score, precision, and recall were compared between models trained and tested on real data versus those involving synthetic data [15], [50], [51]. Specifically, the following scenarios were used to rank performance:

TRTR (Train and Test on Real Data): Baseline performance using real data. TSTR (Train on Synthetic, Test on Real): Measures how well models trained on synthetic data generalize to real data. TRTS (Train on Real, Test on Synthetic): Assesses the consistency of synthetic data in replicating real data behavior. TSTS (Train and Test on Synthetic Data): Evaluates synthetic data's internal consistency and utility within a self-contained model [52], [53]. By comparing these four scenarios, the utility of the synthetic data can be ranked, providing insight into how well it replicates the practical use of real data [54]–[56].

3) **Universal metrices**: Recent research has examined synthetic tabular data via diverse statistical and ML methods [57]–[59]. Nevertheless, these approaches could be more precise regarding suitable evaluation measures. Hence, there's been a move to define universal metrics to boost universality and interpretability. One such metric is TabSynDex, offering a score between 0 and 1 to gauge the synthetic data's closeness to real data [60]. However, this metric still needs to improve in selecting the proper evaluation measures. For example, there needs confirmation that TabSynDex's five main components adequately represent the synthetic data's completeness or facilitate more accessible validation of generated tabular data. A recent SCH Yang et al. proposal introduces a structured evaluation methodology focused on comparing the distributions of synthetic and real data [44]. This study is pivotal as it aligns with generative AI's core principle: replicating the probability distribution of the original dataset.

Therefore, by considering the recent research in the domain, this study employs prominent evaluation metrics such as fidelity (statistical similarity), utility, and class balance in its evaluation section.

III. Synthetic Data Generation Techniques

A. Standard Methods (non-AI)

1) **Random Oversampling (ROS)** is a widely used technique to address class imbalance, a common issue in datasets, especially in domains like cybersecurity where certain types of data, such as network intrusions, are rare compared to normal traffic. ROS works by duplicating samples from the minority class to balance the class distribution, thereby preventing ML models from becoming biased toward the majority class [61]. This technique is particularly relevant when the non-complex small dataset lacks sufficient examples of minority classes.

In synthetic tabular data generation, ROS is often employed to artificially increase the number of minority class samples, helping models better learn the characteristics of these rare events. However, its application in cybersecurity data, which often involves complex relationships between variables, may be limited if used alone, as it does not generate new, diverse data points [62].

The basic procedure of ROS is straightforward:

   a) **Identify the Minority Class**: Determine which class has fewer samples.
   b) **Duplicate Samples**: Randomly select samples from the minority class and duplicate them until the class distribution is balanced.
   c) **Augmented Dataset**: The result is a dataset with a balanced class distribution, where the minority class is equally represented as the majority class

This process does not alter the duplicated samples, meaning the new data points are exact copies of the original samples.

2) **Synthetic Minority Oversampling Technique (SMOTE)** is an advanced over-sampling method designed to address the limitations of ROS by generating synthetic examples rather than merely duplicating existing ones. Introduced by Chawla et al. in 2002, SMOTE aims to balance the class distribution in datasets by creating new instances of the minority class, which helps improve the performance of ML models, particularly in scenarios where the class imbalance is significant [63]. Unlike traditional over-sampling methods, SMOTE synthesizes new samples by interpolating between existing minority class samples, leading to more generalized decision boundaries for classifiers [64].

SMOTE is particularly useful in domains such as cybersecurity, where the occurrence of attacks (minority class) is much rarer than normal behavior (majority class). The technique is well-suited for situations where the minority
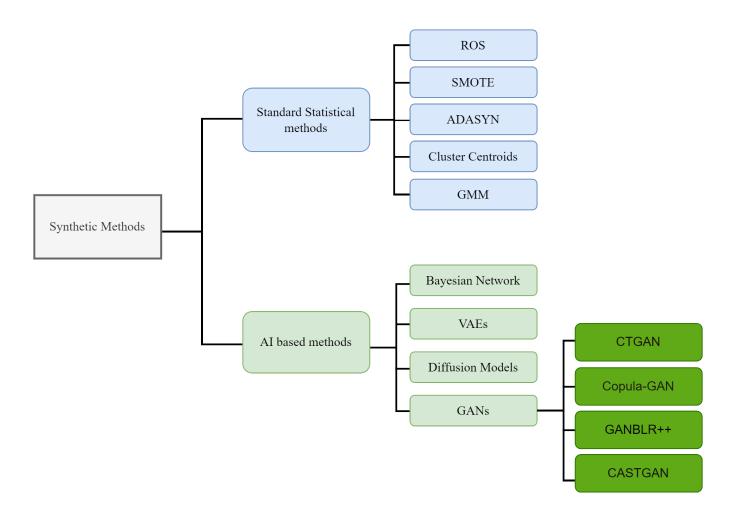
Fig. 1. Methods for generating synthetic tabular data

class instances are spread sparsely across the feature space, and duplicating existing samples (as done in ROS) could lead to overfitting [65], [66].

The SMOTE algorithm operates through the following steps:

a) **Minority Class**: For each minority class instance, there is a need to select k-nearest neighbors, e.g., k = 5

b) **Synthetic Samples**: One of the k-nearest neighbors is randomly chosen to generate a synthetic data point. The synthetic data point is created by interpolating between the feature vector of the original data point and that of the selected neighbor. This interpolation involves calculating the difference between the feature vector of the selected neighbor and the original data point. Then, this difference is multiplied by a random number between 0 and 1, and the resulting value is added to the feature vector of the original data point.

c) **Augmented Dataset**: The result is a dataset augmented with synthetic samples that exhibit varia-

tions within the minority class, thereby mitigating the risk of overfitting and improving the model's ability to generalize.

3) **Adaptive Synthetic Sampling (ADASYN)** is an over-sampling method similar to ROS and SMOTE. However, it introduces a more adaptive approach to generating synthetic data. It was introduced by He et al. in 2008 and focuses on the harder-to-learn examples from the minority class. ADASYN improves on SMOTE by adapting the sampling procedure based on the learning difficulty of each minority class sample. Data points that are harder to classify receive more synthetic neighbors; the rest receive fewer neighbors. [67].

ADASYN is particularly useful in domains such as cybersecurity, where the rarity of specific attack types (minority class) can cause models to perform poorly [68]. By adaptively generating synthetic samples based on the distribution of the minority class, ADASYN helps models learn the decision boundary more effectively, leading to better generalization and improved detection of minority class instances [69].

ADASYN operates through the following steps:

a) **Identify Difficult Minority Class**: ADASYN computes the local density of each minority class by analyzing how many of its nearest neighbors belong to the majority class. Data points surrounded by majority-class neighbors are considered harder to classify.

b) **Weight Assignment**: Weights are assigned to each minority instance/data point based on difficulty level. Harder-to-classify points receive a higher weight, implying that more synthetic samples will be generated.

c) **Synthetic Samples**: For each minority instance, ADASYN interpolates between the instances and one of its nearest minority class neighbors. Synthetic samples are created along the line connecting the two instances.

d) **Data Balancing**: ADASYN adds the generated samples to the original ones, focusing on the problematic region to reduce the class imbalance and improve the model training and learning.

4) **Cluster Centroids** is an under-sampling technique designed to address class imbalance by reducing the number of majority class instances. Unlike traditional under-sampling methods, which randomly remove majority class samples, Cluster Centroids utilizes a clustering algorithm to identify representative centroids within the majority class. The technique was popularized to preserve the majority class's distribution and structure while reducing its size, thereby balancing the dataset without losing critical information [70], [71]. This approach is beneficial in scenarios where the majority class is substantial, and random under-sampling could significantly lose valuable data.

The concept of cluster centroids is applicable in various domains, including cybersecurity. In highly imbalanced datasets, where normal traffic significantly outweighs the attack data, cluster centroids help by grouping similar attack (minority class) samples into clusters and generating synthetic data points at the centroid of each cluster. This helps represent the attack patterns more effectively and exposes ML models to diverse attacks for robust anomaly detection [72].

Cluster Centroids operates through the following steps:

a) **Minority Class Clustering**: Minority class data is grouped into clusters using a clustering algorithm, e.g., K-Means clustering. Each cluster contains similar data points from the minority class.

b) **Centroid Calculation**: For each cluster, the algorithm computes the centroids, the central points representing the average position of the data points within a particular cluster in the feature space.

c) **Synthetic Data**: Synthetic samples are generated at or around the center of each minority cluster. This ensures that the generated data is representative of the overall distribution of the minority class.

d) **Balanced Dataset**: Synthetic data is added to the original data to balance class distributions, improving the AI model's ability to learn from minority classes for higher efficiency.

5) **Gaussian Mixture Model (GMM)** is a probabilistic model used to represent the presence of sub-populations within an overall population without requiring that an instance belong exclusively to a single sub-population. GMMs are particularly useful for modeling data that exhibit multiple underlying distributions, assuming that all data points are generated from a mixture of several Gaussian distributions with unknown parameters [73]. Each Gaussian component within a GMM represents a cluster characterized by its mean, variance, and mixing coefficient.

In the context of synthetic tabular data generation, GMMs approximate the data distribution, making them highly applicable for generating new samples that preserve the statistical properties of the original dataset. This is especially useful in cybersecurity, where modeling the intricate patterns of network traffic or attack vectors requires flexible and accurate representation of complex, multimodal distributions [74]. GMMs provide a robust method for capturing these patterns, allowing for generating synthetic data that closely mimic real-world scenarios [75].

GMM operates through the following steps:

a) **Fitting GMM**: GMM fits the original data by assuming it comes from the mixture of multiple Gaussian distributions. It estimates these Gaussian components' parameters (mean, covariance, and weights) using the Expectation-Maximization (EM) algorithm.

b) **Estimate Gaussian Parameters**: The model assigns data points to different Gaussian components based on their likelihood and calculates the mean (center), covariance (spread), and mixture weight (probability and skewness) for each Gaussian distribution.

c) **Sampling**: After training the model, synthetic data is generated by randomly selecting a Gaussian component (based on its weight) and then sampling a point from the selected Gaussian using its parametric values.

d) **Data Generation**: Repeating the sampling process multiple times generates synthetic data that mimics the structure and underlying distribution of the original data. Thus, the generated data points from all components are combined to form the final synthetic dataset. [76].

B. AI based Methods

1) **Baysian Network (BN)** are probabilistic graphical models that represent a set of variables and their
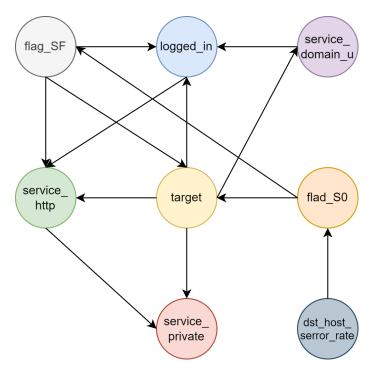
Fig. 2. Bayesian Network illustrating some key feature dependencies in the NSL-KDD dataset

conditional dependencies via a directed acyclic graph (DAG) [77]. Bayesian Networks are particularly effective for modeling complex relationships in data where uncertainty and causality play critical roles [78], [79]. In figure 2, this BN illustrates that `target` is a central feature, influenced by several other variables, including `service_http`, `service_private`, `flag_SF`, `logged_in`, and `flad_S0`. This suggests that `target` plays a crucial role in determining the system's behavior, as it is obvious by its binary data type of normal or attack. The BN also shows key conditional dependencies, such as `flag_SF` being dependent on `logged_in` and `service_domain_u`, while `dst_host_serror_rate` influences `flad_S0`. Some features, like `service_private` and `service_http`, directly impact `target` but do not further influence other variables. Overall, this network emphasizes the significant joint dependencies between these system features, highlighting `target` as a critical node and capturing the complex relationships between the various network-related parameters.

In synthetic data generation, BNs generate data that preserves the inherent dependencies and conditional probabilities observed in the original dataset. This is particularly valuable in cybersecurity, where understanding the intricate dependencies between various network events, user behaviors, or attack patterns is essential for generating realistic synthetic datasets. By capturing these relationships, BNs allow for the generation of synthetic data that not only mimics the distribution of the original data but also maintains the underlying causal

structures, making it highly applicable for scenarios requiring interpretable and robust synthetic data [14]. The process of generating synthetic data using Bayesian Networks involves the following steps [80]:

a) **Structure Learning**: The first step is to determine the structure of the Bayesian Network, which involves identifying the dependencies between variables. This can be done using expert knowledge or algorithms that learn the structure from data, such as hill climbing or score-based methods [81]. Each node in the network corresponds to a variable, and directed edges between nodes represent conditional dependencies.

b) **Parameter Learning**: After defining the structure, the next step is to estimate the conditional probability distributions (CPDs) for each variable in the network, given its parent node. This joint probability information is learned from the original dataset.

c) **Sampling**: After learning the structure and parameters (CPDs), synthetic data is generated by sampling from the Bayesian network. First, the root node is sampled from its marginal distribution, and then the child nodes are sampled based on their conditional probabilities. This process continues throughout the network until all variables are sampled.

d) **Data Generation**: The sampling process is repeated for the required number of synthetic data points. Each sample is a synthesized record for
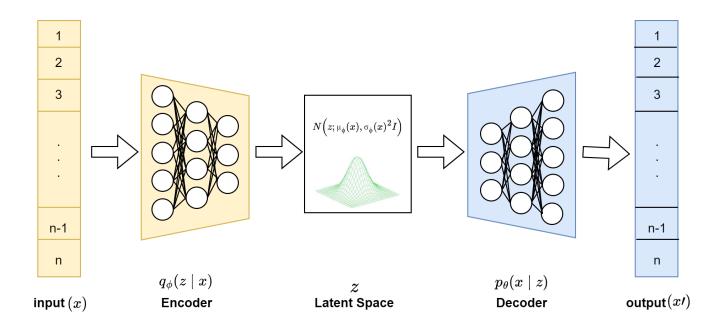
Fig. 3. Variational Autoencoder Structure

the dataset. This process ensures that the generated data depicts the dependencies learned from the original dataset.

2) **Tabular Variational Autoencoder (TVAE)** is a generative model designed to handle the complexities of tabular data, often including a mix of continuous and categorical variables. TVAE has the base idea of autoencoders that consists of two parts: an encoder network that learns to compress high-dimensional data into a low-dimensional, latent spacial representation (the code) and a decoder network that decompresses the compressed representation into the original domain as depicted in Figure 3 [82]. Thus, it extends the traditional Variational Autoencoder (VAE) framework by adapting it to the specific needs of tabular data, ensuring that the generated synthetic data accurately reflects the structure and distribution of the original dataset [83].

Mathematical process of TVAE for Synthetic data generation:

a) **Encoder-Decoder Architecture**: TVAE uses an encoder-decoder structure where the encoder $q_\phi(z|x)$ maps the input data x to a latent space z, and the decoder $p_\theta(x|z)$ reconstructs the data from this latent representation. The encoder outputs the parameters of a Gaussian distribution in the latent space:

$$q_\phi(z|x) = \mathcal{N}(z; \mu_\phi(x), \sigma_\phi(x)^2 I) \qquad (1)$$

Where $\mathcal{N}(z; \mu_\phi(x), \sigma_\phi(x)^2 I)$ represents a multivariate normal (Gaussian) distribution where the latent variable z is assumed to follow a Gaussian distribution with mean $\mu_\phi(x)$ and covariance

matrix $\sigma_\phi(x)^2 I$. The two parameters, $\mu_\phi(x)$ and $\sigma_\phi(x)$, are functions of the input x and are learned by the encoder network. More precisely, $\mu_\phi(x)$ is the mean vector of the Gaussian distribution, parameterized by the input x. This vector represents the expected value of the latent variable z for a given input x, determining the center of the distribution, and $\sigma_\phi(x)^2$ is the variance of the latent variable z, also parameterized by the input x. This variance determines how much the latent variable z can deviate from its mean $\mu_\phi(x)$, controlling the spread of the distribution. Specifically, $\sigma_\phi(x)^2$ is the element-wise square of the standard deviation $\sigma_\phi(x)$. Plus, I is the identity matrix used to represent that the latent variables are assumed to be independent, meaning there is no covariance between different latent dimensions. The use of I ensures that the covariance matrix is diagonal, implying that each latent variable has its independent variance, controlled by $\sigma_\phi(x)^2$.

b) **Variational Objective**: The objective of training TVAE is to maximize the Evidence Lower Bound (ELBO):

$$\mathcal{L}(\theta, \phi; x) = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}(q_\phi(z|x)||p(z)) \qquad (2)$$

Here, $\mathcal{L}(\theta, \phi; x)$ is the objective function (ELBO) to maximize during the training of TVAE. It measures how well the model reconstructs data and regularizes the latent space, and $\log p_\theta(x|z)$ represents the reconstruction loss, ensuring that the generated data $\hat{x}$ resembles the input data x. D_{KL}

is the Kullback-Leibler divergence that regularizes the latent space to follow a prior distribution $p(z)$, typically a standard normal distribution $\mathcal{N}(0, I)$. Plus, $\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)]$ is the expected log-likelihood term, representing the reconstruction loss. It ensures that the generated data $\hat{x}$ resembles the input data x.

  c) **Latent Variable Modeling**: The latent space z is sampled from the Gaussian distribution parameterized by $\mu_\phi(x)$ and $\sigma_\phi(x)$. This latent space effectively captures the dependencies and distribution of the original data.

  d) **Data Generation**: To generate synthetic data, samples are drawn from the latent space z, and these samples are passed through the decoder $p_\theta(x|z)$ to produce synthetic data points $\hat{x}$. The decoder ensures that the synthetic data preserves the statistical characteristics of the original tabular dataset.

  e) **Categorical and Continuous Data Handling**: TVAE incorporates specific mechanisms to handle the peculiarities of tabular data. Standard Gaussian distributions are used for continuous variables, while TVAE often employs a Gumbel-Softmax distribution for categorical variables to allow gradient-based optimization.

3) **Diffusion Models**, particularly Denoising Diffusion Probabilistic Models **(DDPM)**, have emerged as a powerful class of generative models [84], [85]. Originally developed for continuous data types such as images, these models have been adapted to handle the complexities of tabular data, which often includes a mix of categorical and numerical variables. The application of diffusion models to tabular data is exemplified by the **Tabular Diffusion Model (TabDDPM)**, which has shown state-of-the-art performance in generating high-quality synthetic tabular data [86].

Mathematical process of diffusion models for tabular data generation:

  a) **Forward and Reverse Diffusion Processes**: The diffusion model begins with a forward process that gradually adds Gaussian noise to the data. Given a data point x_0, the forward process generates a sequence of increasingly noisy versions $x_1, x_2, \ldots, x_T$, where each step adds a small amount of noise (Figure 4):

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I) \quad (3)$$

Here, β_t is a variance schedule that controls the amount of noise added at each step t. The term $\sqrt{1-\beta_t}$ represents the decay factor, which ensures that the noise doesn't overwhelm the signal, and I is the identity matrix indicating that the noise is isotropic.

The reverse process aims to generate synthetic data by denoising a sample starting from pure noise.

The reverse distribution $p_\theta(x_{t-1}|x_t)$ is parameterized by a neural network that predicts the mean and variance at each step:

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \sigma_\theta^2 I) \quad (4)$$

Here, $\mu_\theta(x_t, t)$ is the predicted mean and σ_θ^2 is the predicted variance at time step t, both parameterized by the neural network θ. The network parameters θ are optimized to minimize the Kullback-Leibler divergence between the forward and reverse processes, effectively learning to denoise the data step by step.

  b) **Loss Function**: The training objective for diffusion models in tabular data is typically the variational lower bound (VLB), which can be simplified to the mean-squared error (MSE) between the predicted noise and the actual noise:

$$\mathcal{L}(\theta) = \mathbb{E}_{t,x_0,\epsilon}\left[\|\epsilon - \epsilon_\theta(x_t, t)\|^2\right] \quad (5)$$

Here, $\epsilon_\theta(x_t, t)$ is the predicted noise by the model at time step t, where x_t represents the noisy data and ϵ_θ is parameterized by the neural network θ. The term ϵ represents the noise added to the data during the forward process. The model minimizes the difference between ϵ and $\epsilon_\theta(x_t, t)$, effectively learning to reverse the noise and reconstruct the data.

The expectation $\mathbb{E}_{t,x_0,\epsilon}$ denotes the averaging over all time steps t, initial data x_0, and noise samples ϵ applied during the forward process.

  c) **Handling Categorical and Numerical Features**: Diffusion models for tabular data, such as TabDDPM, extend the basic framework to handle mixed data types. Categorical variables are typically processed using multinomial diffusion, where the data is represented as one-hot encoded vectors, and the noise is added in a way that respects the unconditional nature of the data:

$$q(x_t|x_{t-1}) = \text{Cat}(x_t; (1-\beta_t)x_{t-1} + \beta_t/K) \quad (6)$$

Here, $\text{Cat}(x_t; \cdot)$ represents the categorical distribution, which ensures that the synthetic data adheres to the unconditional nature of the variable. The term β_t is the variance schedule controlling the amount of noise added, while K represents the number of categories. $(1-\beta_t)x_{t-1} + \beta_t/K$ maintains the correct distribution of categories by adjusting the probability mass of each category at every time step t.

Numerical variables are handled using standard Gaussian diffusion processes. The model learns to denoise both categorical and numerical features in a unified framework, ensuring that the generated synthetic data maintains the relationships and distributions of the original data.
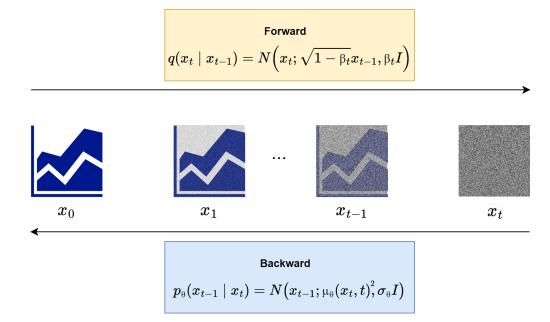
Forward

$$q(x_t \mid x_{t-1}) = N\left(x_t; \sqrt{1 - \beta_t}x_{t-1}, \beta_t I\right)$$

x_0 x_1 ... x_{t-1} x_t

Backward

$$p_\theta(x_{t-1} \mid x_t) = N\left(x_{t-1}; \mu_\theta(x_t, t)^2, \sigma_\theta I\right)$$

Fig. 4. Diffusion Model Flow

d) **Synthetic Data Generation**: After training, the model can generate synthetic tabular data by sampling from the learned distribution. The reverse process generates a sequence of less noisy samples from a sample of pure noise until the final synthetic data point is produced. This approach allows the model to create realistic tabular data that captures individual feature distributions and the dependencies between features.

4) **Generative Adversarial Networks (GAN)**

a) *Conditional Tabular GAN (CTGAN):* CTGAN is a generative model designed explicitly for tabular data, which is often characterized by the presence of mixed data types, including both categorical and continuous variables [12]. CTGAN extends the traditional GAN framework (Figure 5) by incorporating techniques that are well-suited for the unique challenges of tabular data generation, such as handling imbalanced datasets and preserving the relationships between variables [3], [9].

Mathematical process of CTGAN for synthetic data generation:

1) **Generator and Discriminator Networks**: CTGAN uses a generator network $G(z, c)$ that takes as input a noise vector z sampled from a standard normal distribution and a conditional vector c that encodes the categorical features. The generator produces synthetic data samples that mimic the real data distribution. The discriminator network $D(x, c)$ is then tasked with distinguishing between real data samples x and synthetic samples generated by $G(z, c)$ while also considering the conditional vector c.

2) **Mode-Specific Normalization for Continuous Data**: To handle the diverse distribution of continuous features

in tabular data, CTGAN introduces a mode-specific normalization technique. This approach transforms each continuous variable by centering it around a randomly sampled value from its distribution and normalizing it based on the empirical standard deviation of the data within that mode. This process ensures that the generator can more effectively model continuous data distributions:

$$x_{normalized} = \frac{x - x_{sample}}{\sigma(x_{mode})} \quad (7)$$

where x is the original continuous variable, x_{sample} is a value randomly sampled from the distribution of x, and $\sigma(x_{mode})$ is the empirical standard deviation of x within its mode.

3) **Conditional Vector Construction**: CTGAN utilizes a conditional vector c to guide the generation process, particularly for categorical variables. This vector is one-hot encoded, allowing the generator to focus on producing samples that match the specific conditions defined by c. The conditional vector is also integrated into the discriminator's training process, helping it better differentiate between natural and synthetic data based on the specified conditions.

4) **Training Objective**: The training process of CTGAN follows the standard GAN framework, where the generator and discriminator are trained in a min-max game. The generator aims to minimize the following loss function to fool the discriminator, while the discriminator tries to maximize its ability to distinguish real from synthetic data:

$$\min_G \max_D \mathbb{E}_{x \sim p_{data}}[\log D(x, c)] +$$
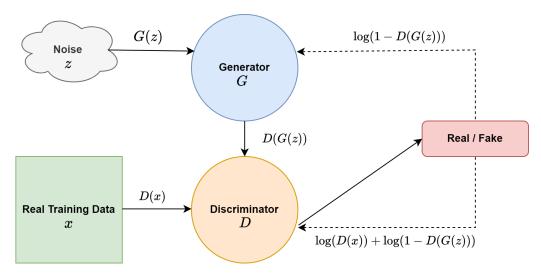$$\mathbb{E}_{z \sim p_z}[\log(1 - D(G(z, c), c))] \quad (8)$$

Fig. 5. Original GAN structure by Goodfellow [9]

Where G represents the generator, D represents the discriminator, x is a data sample drawn from the real data distribution p_{data}, z is a random noise vector drawn from the prior distribution p_z, c denotes any additional conditional information, $G(z, c)$ does the generator generate the synthetic data, and $D(x, c)$ and $D(G(z, c), c)$ are the discriminator's outputs for real and synthetic data, respectively.

5) **Data Sampling and Imbalance Handling**: CTGAN employs a training-by-sampling technique to address class imbalance in tabular data. The data points are sampled during training to ensure underrepresented classes are adequately represented in the training batches. This approach mitigates the impact of imbalanced datasets on the quality of the generated data.

b) Copula GAN: CopulaGAN is a generative model that combines the strengths of copula-based statistical models with the flexibility of GANs to generate synthetic tabular data [87]. Copulas are mathematical tools that describe the dependence structure between random variables, enabling the modeling of complex dependencies separately from the marginal distributions [88]. By integrating copulas into the GAN framework, Copula GAN effectively captures the intricate dependencies in tabular data, making it particularly suitable for scenarios where preserving the relationships between variables is crucial [89].

The mathematical process of Copula-GAN for synthetic data generation:

1) **Marginal Distribution Estimation**: The first step in Copula GAN involves estimating the marginal distributions of each variable in the tabular dataset. This can be achieved using kernel density estimation for continuous variables or empirical distributions for categorical variables. The goal is to transform the original data into a uniform distribution on the interval $[0, 1]$ using the cumulative distribution function (CDF) of each variable:

$$u_i = F_i(x_i) \quad \text{for } i = 1, \ldots, d \tag{9}$$

where u_i is the transformed variable, and $F_i(x_i)$ is the CDF of the i-th variable.

2) **Copula Function for Dependency Modeling**: Copula GAN models the dependencies between the variables using a copula function once the data is transformed into the copula space. The copula function C captures the joint distribution of the transformed variables:

$$C(u_1, \ldots, u_d) = \mathbb{P}(U_1 \le u_1, \ldots, U_d \le u_d) \tag{10}$$

where C denotes the copula function, $u_1, \ldots, u_d$ are the values of the copula-transformed variables, and $U_1, \ldots, U_d$ are the random variables corresponding to these transformed values. The expression $\mathbb{P}(U_1 \le u_1, \ldots, U_d \le u_d)$ represents the joint probability that U_1 is less than or equal to u_1, U_2 is less than or equal to u_2, and so on up to U_d and u_d.

The copula function separates the dependency structure from the marginal distributions, which is advantageous for generating synthetic data that preserves the original data's relationships.

3) **Generator and Discriminator Networks**: In Copula GAN, the generator $G(z)$ takes a noise vector z and generates samples in the copula space, aiming to mimic the dependency structure captured by the copula function. The discriminator $D(u)$ evaluates whether a given sample from the copula space is real or generated, guiding the generator to produce more realistic samples:

$$\min_G \max_D \mathbb{E}_{u \sim p_{data}}[\log D(u)] + \\ \mathbb{E}_{z \sim p_z}[\log(1 - D(G(z)))] \tag{11}$$

4) **Inverse Transformation and Synthetic Data Generation**: After training, the generator produces samples in the copula space. These samples are then transformed

back to the original data space using the inverse of the marginal CDFs:

$$x_i = F_i^{-1}(u_i) \quad \text{for } i = 1, \ldots, d \quad (12)$$

Where x_i represents the i-th component of the synthetic data in the original data space, u_i is the i-th element in the sample in the copula space, and F_i^{-1} denotes the inverse of the marginal CDF of the i-th variable. This inverse transformation ensures that the generated data points follow the original marginal distributions while maintaining the dependencies modeled by the copula.

c) Naive Bayes and Logistic Regression GAN (GAN-BLR++): GANBLR++ is an advanced generative model designed to address the limitations of traditional GANs in generating synthetic tabular data, mainly where complex dependencies and mixed data types (categorical and numerical) are involved. Unlike its predecessor, GANBLR, which primarily focused on modeling categorical data using restricted Bayesian Networks, GANBLR++ introduces significant enhancements that allow it to generate both categorical and numerical data with greater accuracy [14], [90]. This model integrates unrestricted Bayesian Networks and introduces a Dirichlet Mixture Model (DMM) for numerical data generation, enabling it to capture the full range of dependencies in real-world tabular datasets [91]. GANBLR++ is particularly useful in domains where the integrity of the generated data is critical, such as cybersecurity, which offers a robust solution that balances flexibility, accuracy, and computational efficiency.

The mathematical process of GANBLR++ for synthetic data generation:

1) **Incorporation of Unrestricted Bayesian Networks**: GANBLR++ leverages unrestricted Bayesian Networks to model complex dependencies between variables. The network structure is learned through a scoring function that optimizes the log-likelihood of the data while including a regularization term to avoid overfitting. This allows the model to represent more intricate relationships between variables than the restricted networks used in GANBLR.

$$\text{Score}(G : D_{\text{data}}) = \text{LL}(G : D_{\text{data}}) - \lambda \cdot |G| \quad (13)$$

where $\text{Score}(G : D_{data})$ is the score of the network structure G given the data D_{data}, $\text{LL}(G : D_{data})$ denotes the log-likelihood of the data under the network G, λ is the regularization parameter, and $|G|$ represents the complexity of the network G. The regularization term $\lambda \cdot |G|$ helps prevent overfitting by penalizing more complex network structures.

2) **Dirichlet Mixture Model for Numerical Data**: To handle numerical data effectively, GANBLR++ uses a Dirichlet Mixture Model (DMM). The DMM identifies modes within the numerical data distribution and samples from these modes during data generation, ensuring that the generated values accurately reflect the underlying distribution of the original data.

$$x_i \sim \text{N}(\mu_c, \sigma_c) \quad (14)$$

where x_i is the i-th sample from the numerical data, $\text{N}(\mu_c, \sigma_c)$ denotes a Gaussian distribution with mean μ_c and standard deviation σ_c for a particular mode c. The parameters μ_c and σ_c represent the mean and standard deviation of the Gaussian distribution associated with the mode c.

3) **Mode-Specific Sampling**: GANBLR++ further refines the generation of numerical data through mode-specific sampling, which ensures that the generated data points not only match the distributional characteristics of the original data but also maintain the necessary granularity.

$$x_i \sim \text{N}(\mu_c, \sigma_c) \quad \text{subject to}$$
$$\alpha_c < \frac{x_i - \mu_c}{\sigma_c} \leq \alpha_c + 1 \quad (15)$$

where x_i is the i-th sample from the numerical data, $\text{N}(\mu_c, \sigma_c)$ denotes a Gaussian distribution with mean μ_c and standard deviation σ_c for a particular mode c, and α_c is a mode-specific threshold parameter. The constraint $\alpha_c < \frac{x_i - \mu_c}{\sigma_c} \leq \alpha_c + 1$ ensures that x_i falls within a specific range relative to the mode c, maintaining the granularity of the distribution.

d) Cascaded Tabular GAN (CasTGAN): CasTGAN is a novel framework designed for generating realistic tabular data [92]. Traditional GAN models often struggle with capturing the complex dependencies between features in tabular datasets, leading to synthetic data that fails to preserve the validity and interdependencies present in the original data [93]. CasTGAN addresses this limitation by employing a cascaded architecture where each generator in the sequence is responsible for synthesizing a single feature of the dataset. This architecture allows the model to better capture the intricate relationships between features, ensuring that the generated data is not only statistically similar to the real data but also semantically valid. The primary motivation behind CasTGAN is to enhance the generation of high-quality synthetic tabular data, which is crucial in scenarios where data privacy is a concern and real data cannot be freely shared or used.

The mathematical framework of CasTGAN can be broken down into the following steps:

1) **Cascaded Generators:** The model is composed of multiple generators $G_1, G_2, \ldots, G_M$, each responsible for generating one feature of the dataset. Each generator G_i receives inputs from a noise vector z and the outputs of the previous generator G_{i-1}.

2) **Auxiliary Learners:** For each generator G_i, there is an auxiliary learner AL_i that computes a loss specific to the feature generated by G_i. The auxiliary loss guides the generator toward producing more accurate and valid feature values.

3) **Discriminator:** A single discriminator D evaluates the final synthetic dataset, distinguishing between real and

generated data. The discriminator is trained on the output of the final generator G_M, and its loss is propagated back through the cascade to improve the performance of all generators.

4) **Training Objective:** The training of CASTGAN involves minimizing the Wasserstein distance with a gradient penalty between the real and generated data distributions. The value function for the discriminator and the generators is given by:

$$
\begin{aligned}
\min_{G_\to} \max_D V(D) = {}& \mathbb{E}_{x \sim p_{data}}[D(x)] \\
& - \mathbb{E}_{z \sim p_z}[D(G_\to(z))] \\
& - \lambda_{GP} \mathbb{E}_{\hat{x} \sim p_{\hat{x}}} \left[(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1)^2 \right]
\end{aligned}
\tag{16}
$$

where $G_\to$ represents the cascade of generators, D is the discriminator, x is a sample from the real data distribution p_{data}, z is a sample from the prior distribution p_z, $G_\to(z)$ is the generated sample from the cascade of generators, $\hat{x}$ is a sample from the data distribution used for the gradient penalty $p_{\hat{x}}$, $\nabla_{\hat{x}} D(\hat{x})$ is the gradient of the discriminator concerning $\hat{x}$, $\|\nabla_{\hat{x}} D(\hat{x})\|_2$ denotes the L2 norm of this gradient, and λ_{GP} is the gradient penalty coefficient. The term $(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1)^2$ enforces the Lipschitz constraint on the discriminator.

5) **Auxiliary Loss Integration:** Each generator G_i minimizes a combination of the discriminator loss and its corresponding auxiliary loss, scaled by a coefficient λ_{AL_i}.

6) **Final Output:** The final synthetic dataset $\hat{X}$ is generated by passing a noise vector z through the entire cascade of generators.

IV. MUTUAL INFORMATIAON

Mutual Information (MI) measures the mutual dependence between two random variables (linear or non-linear), initially introduced by Shannon in his foundational work on information theory [94]. It quantifies the amount of information gained about one random variable through the observation of another [95]. Unlike the correlation coefficient, limited to real-valued random variables and linear relationships, MI is more general, capturing both linear and non-linear dependencies [96]. The concept of MI is closely related to entropy, a fundamental notion in information theory that measures the expected amount of information in a random variable [97]. MI can be understood as the expected value of the pointwise mutual information (PMI), reflecting how much the joint distribution of two variables deviates from the product of their marginal distributions [98].

A. Comparison with Other Methods

The rationale for selecting MI over other methods lies in its comprehensive mathematical framework. Initially, this study considered two general approaches for choosing the best features based on their dependence on the target feature and their relationships. Correlation was the first and most obvious choice, mathematically defined as the Pearson correlation

coefficient (equ: **??**). However, correlation only accounts for linear relationships [99]. In real-world scenarios, non-linear relationships often need to be identified and quantified to select the optimal set of features.

$$
\mathrm{Corr}(X, Y) = \frac{\mathrm{Cov}(X, Y)}{\sigma_X \sigma_Y}
\tag{17}
$$

where $\mathrm{Cov}(X, Y)$ is the covariance between variables X and Y, and σ_X and σ_Y are the standard deviations of X and Y, respectively. While correlation is effective for capturing linear relationships, it does not account for the non-linear relationships that often exist in real-world data, limiting its effectiveness for feature selection.

The second option was a tree-based AI technique, known for its robustness and ability to identify essential features rigorously [100]. However, tree-based methods suffer from the "black box" problem, offering little interpretability regarding why certain features are selected.

While approaches like Information Gain (IG) have been used for feature selection, as demonstrated by one study [101], Information Gain only measures the relationship between individual features and the target variable, overlooking interdependencies between features [102]. Mathematically, it measures the reduction in entropy when a feature X is used to predict the target variable Y.

$$
IG(Y, X) = H(Y) - H(Y|X)
\tag{18}
$$

where $H(Y)$ is the entropy of the target variable Y, and $H(Y|X)$ is the conditional entropy of Y given X. Although Information Gain quantifies the relationship between individual features and the target variable, it ignores interdependencies among the features.

In contrast, MI captures the relationship between features and the target and the dependencies among the features themselves, providing a more holistic evaluation [103]. MI is defined as:

$$
MI(X, Y) = H(X) + H(Y) - H(X, Y)
\tag{19}
$$

where $H(X)$ and $H(Y)$ are the entropies of variables X and Y, and $H(X, Y)$ is their joint entropy. By capturing non-linear dependencies and interactions among features, MI offers a more comprehensive framework for feature selection.

Compared to other AI-based feature selection methods, such as recursive feature elimination or embedded methods based on decision trees [104], MI has the upper hand by being model-agnostic. Many AI-based techniques are tied to specific algorithms or models, which may introduce biases or limit generalizability across different datasets or classifiers. MI, however, evaluates feature importance based on statistical dependencies independent of any specific model, making it a versatile and unbiased approach [105]. This allows MI to provide a more robust selection process, especially in scenarios where feature relationships are complex and nonlinear, a common occurrence in cybersecurity network data.

B. Application of Mutual Information in this Study

In this study, MI is crucial for measuring the dataset's dependency between different features (variables). It helps understand the strength of relationships between variables, which is essential for modeling and analysis. For example, mutual information was used to quantify the relation between each feature x_i and the target y and between each feature to identify interdependencies within the features. This step was crucial to filter down the essential features that explain the target variables and to cluster the features exhibiting similar behavior.

The mutual information score of every feature was determined concerning the target, and the top 25% of the features were selected based on the mutual information weights (Table III & Table IV). This way, the computation was less intensive, the information was high, and the results were more interpretable.

V. EVALUATION METRICS

The evaluation metrics utilized in this study align with those used in previous studies, as outlined in section II of this paper. The following section details how these metrics were employed in this study and how each metric was measured to assess the experimental results.

1) **Statistical Similarity (Fidelity)** refers to the necessity for synthetic data to resemble the underlying statistical properties of real data closely [106]. This metric's core mathematical definition is that each variable's probability distribution in the synthetic dataset should closely match that of the corresponding variable in the real dataset [44]. However, beyond replicating the individual behavior of each variable, it is equally important to examine the interdependencies and relationships between variables. To evaluate these relationships, we compare the correlations among variables in both datasets [107]. Another crucial aspect is the preservation of the data structure. For instance, if a variable is binary in the real data, it must remain binary in the synthetic data to ensure logical consistency. Similarly, suppose a variable is defined to have values greater than or equal to zero (such as packet size in network traffic data). In that case, the corresponding variable in the synthetic data should also respect this constraint. Failing to maintain these structural properties can lead to synthetic data that, while statistically similar on a superficial level, fails to capture the logical and contextual nuances necessary for accurate downstream analyses and applications [108]. In this study, the following metrics were used to measure the statistical similarity (fidelity), as shown in Tables V to X:

   a) **Data Boundaries (DB):** This metric checks whether the synthetic data adheres to the original data's logical minimum and maximum values. For example, a binary column should contain only binary values, ensuring no out-of-bound entries.

   b) **Correlation (Corr):** This is assessed by examining the correlation heatmap of all variables' absolute values of the correlation coefficients. It evaluates the strength and direction of relationships in both real and synthetic datasets.

   c) **Probability Distribution (PD):** This compares the underlying probability distribution of each variable between the real and synthetic datasets, ensuring that their statistical properties are aligned.

2) **Performance (utility/accuracy)** is a widely used metric to evaluate the utility of synthetic data, particularly in the context of ML and DL models [109]. This metric compares how models trained on synthetic data perform relative to those trained on real data [110]. The most common approach is to contrast TRTR (train and test on real data) with TSTR (train on synthetic data and test on real data). By assessing the model's accuracy, F1 score, precision, and recall, researchers can determine the efficacy of the synthetic data. Suppose the model's performance metrics in the TSTR scenario are comparable to or exceed those in the TRTR scenario. In that case, the synthetic data is considered a viable alternative to the real data [14].

   Moreover, a well-performing TSTR indicates that the synthetic data captures the essential patterns and relationships within the real data, making it suitable for model development, validation, and deployment in privacy-sensitive environments. However, it's crucial to note that synthetic data's utility is not solely dependent on performance metrics; it also relies on the data's ability to generalize across different models and tasks, which should be considered in comprehensive evaluations [111].

   Thus, in this study, the accuracy of TRTR was compared with TSTR for measuring ML utility, as reported in Tables V to X.

3) **Class Balance (CB)** is a critical metric for evaluating the quality of synthetic tabular data, particularly in classification tasks. Class balance refers to the distribution of instances across different classes in the dataset [112]. Maintaining a balanced class distribution in real-world datasets is essential for training robust ML models, as imbalanced courses can lead to biased models that perform poorly on underrepresented classes [113].

   Ensuring that the class distribution closely mirrors the real data is essential when assessing synthetic data. This involves not only comparing the overall proportions of each class but also examining the stability of these proportions across different subsets of the data. Some methods and tools for generating synthetic data inherently include mechanisms to evaluate and maintain class balance, ensuring that the synthetic dataset accurately reflects the class distribution of the real dataset. Suppose a synthetic data generation method does not inherently address class balance. In that case, it may require ad-

ditional post-processing to adjust class distributions and avoid introducing bias into the model training process [114].

Thus, the CB results mentioned in Tables V to X represent how much difference exists between the NORMAL and ATTACK cases. If the classes in the synthetic data are the same, there is zero difference; otherwise, the difference is detailed in the results table later in this paper.

VI. EXPERIMENT SETUP

A. Hardware Setup

The experiments were performed on a workstation with a 13th Gen Intel® Core™ i9-13900 processor. This processor operates at 2000 MHz and features 24 cores with 32 logical processors. The system is equipped with 32 GB of RAM. A dedicated GPU, NVIDIA GeForce RTX 4090, was used for these experiments.

B. Software Setup

The software environment for the experiments was based on Microsoft Windows 11 Pro. The primary programming language used was Python 3.12.4, which provides a stable and versatile platform for implementing various data science and ML techniques. Essential libraries and frameworks include NumPy for numerical operations, Pandas for data manipulation, Seaborn and matplotlib for data visualization, and sci-kit-learn for ML utilities. The experiments also leveraged PyTorch for deep learning tasks. For handling class imbalance, the imbalanced-learn library (learn) was employed. The libraries are crucial for the GAN models, including SDV (Synthetic Data Vault) for tabular data generation, as they support models like CTGAN, TVAE, and CopulaGAN. Due to the lack of pre-existing packages, their official GitHub repositories were also utilized for specialized GAN models such as GANBLR and CASTGAN. PGMPy and SciPy were also used for probabilistic graphical models and scientific computations.

C. Datasets & Pre-processing

1) **NSL-KDD** dataset, a refined version of the original KDD Cup 1999 dataset, is widely used in cybersecurity to evaluate IDS [115]. It addresses some critical issues present in its predecessor, such as redundant records and imbalanced distribution, thereby providing a more accurate and reliable benchmark for performance evaluation. For this study, the training portion of the NSL-KDD dataset, originally consisting of 125,973 instances and 42 columns (41 features and 1 (binary) target), was used. The preprocessing of the NSL-KDD dataset involved several essential steps to prepare it for synthetic data generation and evaluation. Initially, categorical columns were encoded to transform non-numeric data into a suitable format for ML algorithms. This encoding process ensured that all categorical variables were converted into numerical values, facilitating the subsequent analysis. Following the encoding, feature extraction was performed to identify the most significant features using Mutual Information. This process selected the features' top 25% (Q1) based on the mutual information weights. This step resulted in a reduced feature set of 26 encoded features (Table I), which were the most important (based on the information weights; Table III) for accurately representing the data. This refined feature set was then used in the data generation phase, ensuring that the synthetic data closely mirrored the critical characteristics of the original dataset.

2) **CICIDS-17** dataset, part of the Canadian Institute for Cybersecurity's intrusion detection dataset collection, is widely employed for evaluating cybersecurity systems [18]. It provides a comprehensive network traffic data set with diverse attack types and normal traffic, offering a robust benchmark for performance assessment. CICIDS-17 has one week of captured network traffic data in eight comma-separated files (CSV). These files are generated based on the attack types. There are 2,830,743 instances of all eight files and 79 columns (78 numerical features and one binary target).

Preprocessing of the CICIDS-17 dataset involved several vital steps to ready it for synthetic data generation and evaluation. First, all the files were individually tested for null values and duplicated columns. In this process, it was found that there were some infinity values in the data. Thus, all the null values, infinity, and duplicated column *"Fwd Header Length"* were removed. Secondly, all the cleaned data files were concatenated into a single CSV file for further feature selection. Lastly, mutual information was performed to select the variables' top 25% (Q1) based on the mutual information weights. This step resulted in a reduced feature set of 21 features (Table II), which were determined to be the most important (based on the information weights; IV) for accurately representing the data. This reduced feature set was then employed in the synthetic data generation phase, ensuring that the synthetic data accurately reflected the essential characteristics of the original dataset.

VII. EXPERIMENTAL RESULTS AND DISCUSSION

Given the surge in interest in generating synthetic tabular data, the GAN framework has emerged as a seminal approach, attracting substantial research attention. Consequently, this paper delves deeply into evaluating current high-performing GAN models, leveraging recent advancements in this domain. A meticulous analysis of four leading models is presented in Section III, elucidating their methodologies and performance metrics.

A. Results

This study evaluated various methods for generating synthetic tabular data, covering both statistical and AI-based approaches.

TABLE I
SUMMARY OF 26 ENCODED SELECTED FEATURES FROM NSL-KDD DATASET

S.No	Feature	Description
1	src_bytes	Number of data bytes from source to destination
2	dst_bytes	Number of data bytes from destination to source
3	same_srv_rate	Percentage of connections to the same service
4	diff_srv_rate	Percentage of connections to different services
5	flag_SF	Connection status with a normal connection (SF: "Normal")
6	dst_host_srv_count	Number of connections to the same service as the current connection in the past 100 connections
7	dst_host_same_srv_rate	Percentage of connections to the same service for a destination host
8	logged_in	1 if successfully logged in; 0 otherwise
9	dst_host_serror_rate	Percentage of connections that have "SYN" errors
10	dst_host_diff_srv_rate	Percentage of connections to different services for a destination host
11	dst_host_srv_serror_rate	Percentage of connections that have "SYN" errors for a destination host
12	serror_rate	Percentage of connections that have "SYN" errors
13	srv_serror_rate	Percentage of connections that have "SYN" errors for the same service
14	flag_S0	Connection status where no data packets were exchanged (S0: "No Data Exchange")
15	count	Number of connections to the same host as the current connection in the past 2 seconds
16	service_http	HTTP service (1 if used, 0 otherwise)
17	dst_host_srv_diff_host_rate	Percentage of connections to different hosts on the same service
18	level	Threat level of the connection
19	dst_host_count	Number of connections to the same destination host in the past 100 connections
20	dst_host_same_src_port_rate	Percentage of connections with the same source port to the destination host
21	service_private	Private network service (1 if used, 0 otherwise)
22	srv_diff_host_rate	Percentage of connections to different hosts for the same service
23	srv_count	Number of connections to the same service as the current connection in the past 2 seconds
24	dst_host_srv_rerror_rate	Percentage of connections that have "REJ" errors for a destination host
25	service_domain_u	Domain name service (DNS) (1 if used, 0 otherwise)
26	target	Class label indicating if the connection is normal or an attack

1) Statistical Methods: Five widely recognized statistical methods were assessed, as outlined in the survey by [3] (detailed in Section III). The findings indicate the following:

- ROS emerged as the most effective statistical method for preserving the data distribution and ensuring class balance. It maintained a high degree of statistical similarity to the original dataset.
- SMOTE and Cluster Centroid outperformed ROS by maintaining statistical similarity and enhancing ML utility, achieving a 99% accuracy while preserving class balance.
- GMM showed strong performance in capturing feature correlations but underperformed in statistical similarity and ML utility, with an accuracy of only 53%.
- The results were consistent across both the synthetic NSL-KDD and CICIDS-17 datasets (Table V and Table VI).

2) AI Methods: Four prominent AI and generative AI methodologies were evaluated (details in Section III). The results are summarized as follows:

- Bayesian Networks struggled with maintaining correlation and structure in the NSL-KDD dataset, especially for binary features. Diffusion models had the lowest ML utility accuracy at 51%.

- TVAE and Conditional Tabular CTGAN performed exceptionally well, preserving statistical properties and achieving high ML utility accuracy rates of 98% and 96%, respectively (Table VII).
- Consistent performance was observed with the CICIDS-17 dataset, where TVAE slightly outperformed CTGAN in ML utility.

3) Prominent GANs: Four leading GAN models were evaluated in depth (detailed in Section III). The results showed:

- CTGAN and Copula-GAN outperformed other models, with ML utility accuracy scores of 97% and 98%, respectively, on the NSL-KDD and CICIDS-17 datasets.
- CastGAN showed strong performance in handling feature dependencies but struggled to preserve boundary conditions for specific binary columns.
- GANBLR++, based on Bayes' network principles, exhibited performance metrics closely aligned with those of individual BN models.

B. Discussion

The results from this study provide several key insights into the effectiveness of various synthetic data generation methods:

1) Statistical Methods: While effective at maintaining data distribution, ROS is limited in generating genuinely synthetic

S.No	Feature	Description
1	Average Packet Size	Average size of the packets in the flow
2	Packet Length Std	Standard deviation of the packet lengths in the flow
3	Packet Length Variance	Variance of the packet lengths in the flow
4	Packet Length Mean	Mean length of the packets in the flow
5	Total Length of Bwd Packets	Total number of bytes in backward (Bwd) packets
6	Subflow Bwd Bytes	Number of bytes in the backward sub-flow
7	Destination Port	Port number of the destination host
8	Avg Bwd Segment Size	Average size of backward segments
9	Bwd Packet Length Mean	Mean length of backward packets
10	Init_Win_bytes_forward	Initial window size in bytes for forward direction
11	Subflow Fwd Bytes	Number of bytes in the forward sub-flow
12	Total Length of Fwd Packets	Total number of bytes in forward (Fwd) packets
13	Max Packet Length	Maximum length of a packet in the flow
14	Bwd Packet Length Max	Maximum length of a backward packet
15	Init_Win_bytes_backward	Initial window size in bytes for backward direction
16	Fwd Packet Length Max	Maximum length of a forward packet
17	Fwd Packet Length Mean	Mean length of forward packets
18	Avg Fwd Segment Size	Average size of forward segments
19	Flow IAT Max	Maximum time interval between packets in the flow
20	Flow Bytes/s	Rate of flow in bytes per second
21	target	Class label indicating if the flow is benign or malicious

TABLE III
MUTUAL INFORMATION SCORE OF NSL-KDD FEATURES

Feature	Mutual Information Score
src_bytes	0.566864
dst_bytes	0.439281
same_srv_rate	0.369288
diff_srv_rate	0.361895
flag_SF	0.341828
dst_host_srv_count	0.335993
dst_host_same_srv_rate	0.309832
logged_in	0.292075
dst_host_serror_rate	0.286589
dst_host_diff_srv_rate	0.283874
dst_host_srv_serror_rate	0.281332
serror_rate	0.278666
srv_serror_rate	0.269186
flag_S0	0.263399
count	0.262733
service_http	0.191343
dst_host_srv_diff_host_rate	0.189822
level	0.153819
dst_host_count	0.144479
dst_host_same_src_port_rate	0.131316
service_private	0.118493
srv_diff_host_rate	0.099441
srv_count	0.062476
dst_host_srv_rerror_rate	0.062244
service_domain_u	0.048430

TABLE IV
MUTUAL INFORMATION SCORE OF CICIDS-17 FEATURES

Feature	Mutual Information Score
Average Packet Size	0.347112
Packet Length Std	0.342188
Packet Length Variance	0.342019
Packet Length Mean	0.319635
Total Length of Bwd Packets	0.296955
Subflow Bwd Bytes	0.296882
Destination Port	0.291245
Avg Bwd Segment Size	0.287676
Bwd Packet Length Mean	0.287483
Init_Win_bytes_forward	0.287174
Subflow Fwd Bytes	0.284528
Total Length of Fwd Packets	0.284264
Max Packet Length	0.264060
Bwd Packet Length Max	0.263210
Init_Win_bytes_backward	0.250188
Fwd Packet Length Max	0.246537
Fwd Packet Length Mean	0.213223
Avg Fwd Segment Size	0.213065
Flow IAT Max	0.212817
Flow Bytes/s	0.209784

fundamental definition of samples from fitted distributions. Still, its lower utility score suggests it may not be ideal for all applications [121], [122].

2) AI Methods: Bayesian Networks' struggle with larger datasets, particularly with multi-class features, indicates a need for better handling such data structures in AI models. The potential reason for this struggle comes from the probabilistic graphical model. This graph tries to represent dependency between variables, and in the case of the CIC-IDS large dataset, it fails to create discretization because of low representation learning (memory) [123], [124]. Diffusion models require further refinement to improve their statistical similarity and utility because they are, in general, designed for

data, making it less suitable for cybersecurity applications. By definition, ROS does not generate novel data points; it is just random sampling. SMOTE and Cluster Centroid, by enhancing ML utility, present better alternatives for applications where both statistical fidelity and predictive accuracy are critical. Both SMOTE and cluster centroids create novel data points by interpolation and average representation, respectively [63], [120]. GMM's performance highlights its strength in capturing feature correlations and creating novel data points using the

TABLE V
COMPARISON OF SYNTHETIC DATA BY STANDARD METHODS ON NSL-KDD DATASET

Method	Description	Pros	Cons	Performance Summary				
				SS			CB	Accuracy
				DS	Corr	PD		
ROS [116]	Randomly duplicates minority class samples	Simple, low cost	Overfitting risk, no new samples	Yes	Yes	0% diff	0% diff	0.9998
SMOTE [63]	Generates synthetic samples by interpolation	Reduces overfitting, better minority representation	May create ambiguous samples, high computation	No	Yes	0% diff	0% diff	0.9999
Adaptive Synthetic Sampling [67]	Adjusts sampling based on data distribution	Flexible, effective for imbalance	Complex tuning, high computation	No	Yes	3.8% diff	0.14% diff	0.9998
Cluster based oversampling [117]	Oversamples within identified clusters	Captures local structures	Sensitive to clustering algorithm	No	Yes	0% diff	0% diff	0.9995
Gaussian Mixture Model [75]	Uses Gaussian distributions for sampling	Captures complex distributions	Assumes data distribution	No	Yes	99.1% diff	99.9% diff	0.5314

"**SS**" = Statistical Similarity, "**DS**" = Data Structure, "**Corr**" = Correlation, "**PD**" = Probability Distribution, "**CB**" = Class Balance, "**Accuracy**" = Machine Learning Utility on TRTR vs TSTR, "**TRTR**" = Train and Test on Real Data = **0.9995**, "**TSTR**" = Train on Synthetic and Test on Real Data

TABLE VI
COMPARISON OF SYNTHETIC DATA BY STANDARD METHODS ON CICIDS-17 DATASET

Method	Statistical Similarity			Class Balance	Accuracy (**0.9978**)
	Data Structure	Correlation	Probability Distribution		
ROS	Yes	Yes	0% diff	0% diff	0.9991
SMOTE	Yes	Yes	0% diff	0% diff	0.9990
Adaptive Synthetic Sampling	Yes	Yes	0% diff	0.2% diff	0.9986
Cluster-based oversampling	No	Yes	0% diff	0% diff	0.9930
Gaussian Mixture Model	No	No	61.9% diff	46.1% diff	0.4509

"**Accuracy**" = Machine Learning Utility on TRTR vs TSTR, "**TRTR**" = Train and Test on Real Data, "**TSTR**" = Train on Synthetic and Test on Real Data

high-dimensional data with spatial relationships (e.g., images), making it challenging to capture the non-spatial, relational structure of tabular data [85]. The success of TVAE and CTGAN emphasizes the growing importance of generative AI models in synthetic data generation. However, their lack of inherent class imbalance solutions suggests that additional techniques may be required to address this issue comprehensively.

TVAE and CTGAN excel in generating high-quality synthetic tabular data due to their specialized architectures and loss functions. TVAE uses a probabilistic latent space to model complex dependencies, handling continuous and categorical features with different probability distributions [125]. Its KL divergence-based loss regularizes the model, ensuring diverse yet realistic data generation [126]. CTGAN, on the other hand, uses a conditional generator to capture feature relationships and a specific loss function that optimizes for mode-specific normalization, preserving the distributions of both majority and minority categories [127]. The combination of these tailored architectures and well-designed loss functions enables both models to generate synthetic data that closely mirrors the statistical properties of the original dataset, making them

highly effective for tabular data generation [12].

3) Prominent GANs: The superior performance of CTGAN and Copula-GAN reinforces their suitability for generating synthetic tabular data that retains statistical integrity and practical utility. CopulaGAN excels in generating synthetic tabular data because it uses copulas to model the dependencies between variables and their marginal distributions separately [128]. This allows it to capture complex relationships between features more accurately than standard GANs. By decoupling the modeling of individual feature distributions from their joint dependency structure, CopulaGAN ensures that the synthetic data maintains the statistical properties of the original dataset, making it highly effective for tabular data [129]. CastGAN's innovative architecture of each generator and auxiliary learner for each variable shows promise, though its limitations with binary features suggest areas for future improvement [92]. The effectiveness of GANBLR++ demonstrates the value of integrating Bayesian principles into GAN frameworks, providing a robust foundation for capturing complex probabilistic relationships in the data for both numerical and categorical datasets [90].

TABLE VII

COMPARISON OF SYNTHETIC DATA BY AI METHODS ON NSL-KDD DATASET

| Method | Description | Pros | Cons | Performance Summary | | | | |
| | | | | SS | | | CB | Accuracy |
				DS	Corr	PD		
Bayesian Neural Networks (BN) [118]	Uses Bayesian inference for uncertainty estimation	Quantifies uncertainty, probabilistic dependencies	Requires prior knowledge, complex tuning	No	No	0% diff	17.2% diff	0.9587
Variational Tabular Autoencoders (TVAEs) [119]	Neural networks for unsupervised feature learning	Learns representations, reduces dimensionality	KL divergence issue, risk of overfitting	Yes	Yes	23.1% diff	26.7% diff	0.9807
Diffusion Models [86]	Uses diffusion for data generation	Captures temporal dependencies	Needs sequential data, high computational cost	No	No	98.3% diff	34.2% diff	0.51
Conditional Tabular Generative Adversarial Networks (CTGAN) [12]	Adversarial training between generator and discriminator	High-quality data, models complex distributions	Training instability, mode collapse	Yes	Yes	11.5% diff	5.7% diff	0.9640

SS = Statistical Similarity, **DS** = Data Structure, **Corr** = Correlation, **PD** = Probability Distribution, **CB** = Class Balance, "**Accuracy**" Machine Learning Utility on TRTR vs TSTR, "**TRTR**" = Train and Test on Real Data = **0.9995**, "**TSTR**" = Train on Synthetic and Test on Real Data

TABLE VIII

COMPARISON OF SYNTHETIC DATA BY AI METHODS ON CICIDS-17 DATASET

| Method | Statistical Similarity | | | Class Balance | Accuracy (**0.9978**) |
	Data Structure	Correlation	Probability Distribution		
Bayesian Neural Networks (BN)	–	–	–	–	–
Tabular Variational Autoencoder (TVAE)	Yes	No	1.4% diff	69.4% diff	0.9718
Conditional Tabular GAN (CTGAN)	Yes	Yes	0.8% diff	5.2% diff	0.9603

"**Accuracy**" Machine Learning Utility on TRTR vs TSTR, "**TRTR**" = Train and Test on Real Data, "**TSTR**" = Train on Synthetic and Test on Real Data

TABLE IX

COMPARISON OF SYNTHETIC DATA BY PROMINENT GANs ON NSL-KDD DATASET

| Method | Statistical Similarity | | | Class Balance | Accuracy (**0.9995**) |
	Data Structure	Correlation	Probability Distribution		
CTGAN	Yes	Yes	7.7% diff	6.7% diff	0.9667
GANBLR++	No	No	99% diff	8.8% diff	0.5916
CopulaGAN	Yes	Yes	7.7% diff	5.4% diff	0.9761
CastGAN	No	Yes	27% diff	11.1% diff	0.9582

"**Accuracy**" Machine Learning Utility on TRTR vs TSTR, "**TRTR**" = Train and Test on Real Data, "**TSTR**" = Train on Synthetic and Test on Real Data

VIII. FUTURE WORK

Building upon the insights derived from this study, potential directions for future investigations include:

1) **Real vs AI class balance**: A predominant challenge in ML tasks is ensuring balanced classes for unbiased predictions. This requirement raises questions about the real-world balance of attack classes, necessitating thorough examination. It is crucial to scrutinize whether achieving balance among ML classes facilitates accurate data synthesis and optimizes ML tasks.

2) **Benchmarking the boundary of attack and normal traffic**: This research applies generative AI techniques to analyze the distinction between normal and abnormal network traffic rigorously. Through this exploration, we aim to enhance our comprehension of threshold dynamics, significantly contributing to developing more resilient IDS.

3) **Fidelity vs Privacy**: Enhancing the fidelity and utility of synthetic data often poses privacy risks. Thus, it is imperative to identify an optimized balance between fidelity and privacy, particularly concerning cybersecurity datasets. This endeavor requires careful consideration

TABLE X
COMPARISON OF SYNTHETIC DATA BY PROMINENT GANS ON CICIDS-17 DATASET

Method	Statistical Similarity			Class Balance	Accuracy (**0.9978**)
	Data Structure	Correlation	Probability Distribution		
CTGAN	Yes	Yes	0% diff	5% diff	0.95381
GANBLR++	No	No	47.6% diff	64.4% diff	0.5378
CopulaGAN	Yes	Yes	0% diff	5% diff	0.9755
CastGAN	No	No	4.8% diff	58.3% diff	0.9734

"**Accuracy**" Machine Learning Utility on TRTR vs TSTR, "**TRTR**" = Train and Test on Real Data, "**TSTR**" = Train on Synthetic and Test on Real Data

to safeguard individual privacy while maintaining the data's utility for cybersecurity purposes.

4) **Data dynamic features**: Develop and refine feature selection methods targeting data dynamic features in network flows. These techniques should adapt to evolving network environments, identifying the most relevant features for real-time IDS.

5) **Adaptive IDS**: Design IDS that dynamically adjusts to changes in network traffic patterns. This would involve real-time analysis and feature selection to detect new threats emerging as the network evolves.

6) **Loss functions**: Conduct a comparative analysis of various generative AI loss functions tailored to the study's specific context. This will help identify which loss functions are most effective in optimizing the application's model performance, providing insights into their strengths and limitations in different scenarios.

7) **Correlation vs. Dependency**: To examine the deeper relationships between variables by exploring logical and conditional dependencies, going beyond the surface-level insights provided by simple correlation measures. This will allow a more critical understanding of how variables interact, particularly in complex datasets where correlations may not fully capture the underlying dependencies.

IX. CONCLUSION

Our comparative study systematically explores the effectiveness of synthetic tabular data generation techniques within cybersecurity network traffic, covering both daily and malicious traffic scenarios. Through a detailed evaluation of several synthetic data generation methods across two significant datasets, we have revealed critical insights into their performances and the inherent trade-offs they entail. SMOTE and Cluster Centroids demonstrated substantial data similarity and class balance but suffer from the drawback of producing redundant, non-novel values. Conversely, CTGAN and TVAE outperformed the other methods, although they do not inherently ensure balanced class distribution. CastGAN emerges as a strong contender by maintaining internal feature dependencies, offering a more balanced approach. The insights from this study provide essential guidance for selecting synthetic data generation techniques based on specific cybersecurity requirements. This research significantly impacts the field by highlighting the strengths and limitations of each method, offering a clear framework for optimizing synthetic data generation in cybersecurity applications. The source code of this experiment is available at our https://github.com/AdenRajput/Comparative_Analysis.git.

X. ACKNOWLEDGEMENTS

REFERENCES

[1] S. Layeghy, M. Gallagher, and M. Portmann, "Benchmarking the benchmark—comparing synthetic and real-world network ids datasets," *Journal of Information Security and Applications*, vol. 80, p. 103689, 2024.

[2] A. Dunmore, J. Jang-Jaccard, F. Sabrina, and J. Kwak, "A comprehensive survey of generative adversarial networks (gans) in cybersecurity intrusion detection," *IEEE Access*, 2023.

[3] A. Figueira and B. Vaz, "Survey on synthetic data generation, evaluation methods and gans," *Mathematics*, vol. 10, no. 15, p. 2733, 2022.

[4] 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.

[5] E. Espinosa and A. Figueira, "On the quality of synthetic generated tabular data," *Mathematics*, vol. 11, no. 15, p. 3278, 2023.

[6] T. Sattarov, M. Schreyer, and D. Borth, "Findiff: Diffusion models for financial tabular data generation," in *Proceedings of the Fourth ACM International Conference on AI in Finance*, pp. 64–72, 2023.

[7] R. McDowall, "Data integrity focus, part ix: Understanding data integrity metrics for chromatography," 2019.

[8] E. M. Bordewijk, R. Wang, L. M. Askie, L. C. Gurrin, J. G. Thornton, M. van Wely, W. Li, and B. W. Mol, "Data integrity of 35 randomised controlled trials in women'health," 2020.

[9] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[10] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *International conference on machine learning*, pp. 214–223, PMLR, 2017.

[11] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," *Advances in neural information processing systems*, vol. 30, 2017.

[12] L. Xu, M. Skoularidou, A. Cuesta-Infante, and K. Veeramachaneni, "Modeling tabular data using conditional gan," *Advances in neural information processing systems*, vol. 32, 2019.

[13] S. Kamthe, S. Assefa, and M. Deisenroth, "Copula flows for synthetic data generation," *arXiv preprint arXiv:2101.00598*, 2021.

[14] Y. Zhang, N. Zaidi, J. Zhou, and G. Li, "Interpretable tabular data generation," *Knowledge and Information Systems*, vol. 65, no. 7, pp. 2935–2963, 2023.

[15] S. Bourou, A. El Saer, T.-H. Velivassaki, A. Voulkidis, and T. Zahariadis, "A review of tabular data synthesis using gans on an ids dataset," *Information*, vol. 12, no. 09, p. 375, 2021.

[16] D. Ganji and C. Chakraborttii, "Towards data generation to alleviate privacy concerns for cybersecurity applications," in *2023 IEEE 47th Annual Computers, Software, and Applications Conference (COMPSAC)*, pp. 1447–1452, IEEE, 2023.

[17] A. Kotal, A. Piplai, S. S. L. Chukkapalli, and A. Joshi, "Privetab: Secure and privacy-preserving sharing of tabular data," in *Proceedings of the 2022 ACM on International Workshop on Security and Privacy Analytics*, pp. 35–45, 2022.

[18] I. Sharafaldin, A. H. Lashkari, A. A. Ghorbani, *et al.*, "Toward generating a new intrusion detection dataset and intrusion traffic characterization.," *ICISSp*, vol. 1, pp. 108–116, 2018.

[19] V. Dixit, R. Selvarajan, T. Aldwairi, Y. Koshka, M. A. Novotny, T. S. Humble, M. A. Alam, and S. Kais, "Training a quantum annealing based restricted boltzmann machine on cybersecurity data," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 6, no. 3, pp. 417–428, 2021.

[20] Z. Zhong, C. Xie, and X. Tang, "Intrusion traffic detection and classification based on unsupervised learning," *IEEE Access*, 2024.

[21] T. Kim and W. Pak, "Early detection of network intrusions using a gan-based one-class classifier," *IEEE Access*, vol. 10, pp. 119357–119367, 2022.

[22] I. H. Sarker, A. Kayes, S. Badsha, H. Alqahtani, P. Watters, and A. Ng, "Cybersecurity data science: an overview from machine learning perspective," *Journal of Big data*, vol. 7, pp. 1–29, 2020.

[23] S. Greengard, "Cybersecurity gets smart," *Communications of the ACM*, vol. 59, no. 5, pp. 29–31, 2016.

[24] M. Chalé and N. D. Bastian, "Generating realistic cyber data for training and evaluating machine learning classifiers for network intrusion detection systems," *Expert Systems with Applications*, vol. 207, p. 117936, 2022.

[25] Y. N. Rao and K. Suresh Babu, "An imbalanced generative adversarial network-based approach for network intrusion detection in an imbalanced dataset," *Sensors*, vol. 23, no. 1, p. 550, 2023.

[26] Z. Lin, Y. Shi, and Z. Xue, "Idsgan: Generative adversarial networks for attack generation against intrusion detection," in *Pacific-asia conference on knowledge discovery and data mining*, pp. 79–91, Springer, 2022.

[27] J. Vaidya and C. Clifton, "Privacy-preserving data mining: Why, how, and when," *IEEE Security & Privacy*, vol. 2, no. 6, pp. 19–27, 2004.

[28] S. Rodda and U. S. R. Erothi, "Class imbalance problem in the network intrusion detection systems," in *2016 international conference on electrical, electronics, and optimization techniques (ICEEOT)*, pp. 2685–2688, Ieee, 2016.

[29] V. Torra and Y. Narukawa, "Special issue on data mining and data privacy," *Journal of Ambient Intelligence and Humanized Computing*, vol. 14, no. 11, pp. 14977–14978, 2023.

[30] D.-W. Kim, G.-Y. Sin, K. Kim, J. Kang, S.-Y. Im, and M.-M. Han, "Network traffic synthesis and simulation framework for cybersecurity exercise systems," 2024.

[31] D.-W. Kim, G.-Y. Shin, Y.-H. Jang, S. Cho, K. Kim, J. Kang, and M.-M. Han, "Framework for network topology generation and traffic prediction analytics for cyber exercises," *IEEE Access*, 2023.

[32] V. S. Chundawat, A. K. Tarun, M. Mandal, M. Lahoti, and P. Narang, "Tabsyndex: a universal metric for robust evaluation of synthetic tabular data," *arXiv preprint arXiv:2207.05295*, 2022.

[33] X. Li, L. Khan, M. Zamani, S. Wickramasuriya, K. Hamlen, and B. Thuraisingham, "Con2mix: A semi-supervised method for imbalanced tabular security data 1," *Journal of Computer Security*, no. Preprint, pp. 1–22, 2023.

[34] F. Yan, S. Wen, S. Nepal, C. Paris, and Y. Xiang, "Explainable machine learning in cybersecurity: A survey," *International Journal of Intelligent Systems*, vol. 37, no. 12, pp. 12305–12334, 2022.

[35] C. Han and R. Xue, "Differentially private gans by adding noise to discriminator's loss," *Computers & Security*, vol. 107, p. 102322, 2021.

[36] C. Yinka-Banjo and O.-A. Ugot, "A review of generative adversarial networks and its application in cybersecurity," *Artificial Intelligence Review*, vol. 53, pp. 1721–1736, 2020.

[37] A. Dunmore, J. Jang-Jaccard, F. Sabrina, and J. Kwak, "A comprehensive survey of generative adversarial networks (gans) in cybersecurity intrusion detection," *IEEE Access*, vol. 11, pp. 76071–76094, 2023.

[38] R. Mohammad, F. Saeed, A. A. Almazroi, F. S. Alsubaei, and A. A. Almazroi, "Enhancing intrusion detection systems using a deep learning and data augmentation approach," *Systems*, vol. 12, no. 3, p. 79, 2024.

[39] A. Kothare, S. Chaube, Y. Moharir, G. Bajodia, and S. Dongre, "Syngen: synthetic data generation," in *2021 International Conference on Computational Intelligence and Computing Applications (ICCICA)*, pp. 1–4, IEEE, 2021.

[40] R. Mayer, M. Hittmeir, and A. Ekelhart, "Privacy-preserving anomaly detection using synthetic data," in *Data and Applications Security and Privacy XXXIV: 34th Annual IFIP WG 11.3 Conference, DBSec 2020, Regensburg, Germany, June 25–26, 2020, Proceedings 34*, pp. 195–207, Springer, 2020.

[41] K. El Emam, L. Mosquera, and J. Bass, "Evaluating identity disclosure risk in fully synthetic health data: model development and validation," *Journal of medical Internet research*, vol. 22, no. 11, p. e23139, 2020.

[42] G. Agrawal, A. Kaur, and S. Myneni, "A review of generative models in generating synthetic attack data for cybersecurity," *Electronics*, vol. 13, no. 2, p. 322, 2024.

[43] J. Yoon, D. Jarrett, and M. Van der Schaar, "Time-series generative adversarial networks," *Advances in neural information processing systems*, vol. 32, 2019.

[44] S. C.-H. Yang, B. Eaves, M. T. Schmidt, K. B. Swanson, and P. Shafto, "Structured evaluation of synthetic tabular data," 2023.

[45] D. Foster, *Generative deep learning*. " O'Reilly Media, Inc.", 2022.

[46] K. T. Ahmed, S. Aslam, H. Afzal, S. Iqbal, A. Mehmood, and G. S. Choi, "Symmetric image contents analysis and retrieval using decimation, pattern analysis, orientation, and features fusion," *IEEE Access*, vol. 9, pp. 57215–57242, 2021.

[47] D. Fuentes, D. McSpadden, and S. Adewole, "Distributed conditional gan (discgan) for synthetic healthcare data generation," *arXiv preprint arXiv:2304.04290*, 2023.

[48] H. Thimonier, F. Popineau, A. Rimmel, and B.-L. Doan, "Beyond individual input for deep anomaly detection on tabular data," *arXiv preprint arXiv:2305.15121*, 2023.

[49] V. Bindschaedler and R. Shokri, "Synthesizing plausible privacy-preserving location traces," in *2016 IEEE Symposium on Security and Privacy (SP)*, pp. 546–563, IEEE, 2016.

[50] N. Peppes, T. Alexakis, K. Demestichas, and E. Adamopoulou, "A comparison study of generative artificial network architectures for malicious cyber-attack data generation," *Applied Sciences*, vol. 13, no. 12, p. 7106, 2023.

[51] A. Ceccarelli and T. Zoppi, "Intrusion detection without attack knowledge: generating out-of-distribution tabular data," in *2023 IEEE 34th International Symposium on Software Reliability Engineering (ISSRE)*, pp. 125–136, IEEE, 2023.

[52] S. Towhidul Islam Tonmoy and S. Mehedi Zaman, "Oog-optuna optimized gan sampling technique for tabular imbalanced malware data," *arXiv e-prints*, pp. arXiv–2212, 2022.

[53] R. A. Nugraha, H. F. Pardede, and A. Subekti, "Oversampling based on generative adversarial networks to overcome imbalance data in predicting fraud insurance claim: 10.48129/kjs. splml. 19119," *Kuwait Journal of Science*, 2022.

[54] N. Peppes, T. Alexakis, E. Adamopoulou, and K. Demestichas, "The effectiveness of zero-day attacks data samples generated via gans on deep learning classifiers," *Sensors*, vol. 23, no. 2, p. 900, 2023.

[55] A. Hu, R. Xie, Z. Lu, A. Hu, and M. Xue, "Tablegan-mca: Evaluating membership collisions of gan-synthesized tabular data releasing," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pp. 2096–2112, 2021.

[56] E. Lomurno, A. Archetti, L. Cazzella, S. Samele, L. Di Perna, and M. Matteucci, "Sgde: Secure generative data exchange for cross-silo federated learning," in *Proceedings of the 2022 5th International Conference on Artificial Intelligence and Pattern Recognition*, pp. 205–214, 2022.

[57] Z. Zhao, R. Birke, and L. Y. Chen, "Fct-gan: Enhancing global correlation of table synthesis via fourier transform," in *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pp. 4450–4454, 2023.

[58] A. J. Rodriguez-Almeida, H. Fabelo, S. Ortega, A. Deniz, F. J. Balea-Fernandez, E. Quevedo, C. Soguero-Ruiz, A. M. Wägner, and G. M. Callico, "Synthetic patient data generation and evaluation in disease prediction using small and imbalanced datasets," *IEEE Journal of Biomedical and Health Informatics*, 2022.

[59] R. Rai and S. Sural, "Tool/dataset paper: Realistic abac data generation using conditional tabular gan," in *Proceedings of the Thirteenth ACM Conference on Data and Application Security and Privacy*, pp. 273–278, 2023.

[60] V. S. Chundawat, A. K. Tarun, M. Mandal, M. Lahoti, and P. Narang, "A universal metric for robust evaluation of synthetic tabular data," *IEEE Transactions on Artificial Intelligence*, 2022.

[61] S. Bagui and K. Li, "Resampling imbalanced data for network intrusion detection datasets," *Journal of Big Data*, vol. 8, no. 1, p. 6, 2021.

[62] A. Kaur and M. Sarmadi, "Comparative analysis of machine learning techniques for imbalanced genetic data," *Annals of Data Science*, pp. 1–23, 2024.

[63] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[64] L. Lusa *et al.*, "Evaluation of smote for high-dimensional class-imbalanced microarray data," in *2012 11th international conference on machine learning and applications*, vol. 2, pp. 89–94, IEEE, 2012.

[65] J. Sun, J. Lang, H. Fujita, and H. Li, "Imbalanced enterprise credit evaluation with dte-sbd: Decision tree ensemble based on smote and bagging with differentiated sampling rates," *Information Sciences*, vol. 425, pp. 76–91, 2018.

[66] J. Sun, H. Li, H. Fujita, B. Fu, and W. Ai, "Class-imbalanced dynamic financial distress prediction based on adaboost-svm ensemble combined with smote and time weighting," *Information Fusion*, vol. 54, pp. 128–144, 2020.

[67] H. He, Y. Bai, E. A. Garcia, and S. Li, "Adasyn: Adaptive synthetic sampling approach for imbalanced learning," in *2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence)*, pp. 1322–1328, Ieee, 2008.

[68] A. S. Hussein, T. Li, D. M. Abd Ali, K. Bashir, and C. W. Yohannese, "A modified adaptive synthetic sampling method for learning imbalanced datasets," in *Developments of Artificial Intelligence Technologies in Computation and Robotics: Proceedings of the 14th International FLINS Conference (FLINS 2020)*, pp. 76–83, World Scientific, 2020.

[69] A. S. A. AL-Ghamdi, M. Ragab, M. F. S. Sabir, A. Elhassanein, and A. A. Gouda, "Optimized artificial neural network techniques to improve cybersecurity of higher education institution.," *Computers, Materials & Continua*, vol. 72, no. 2, 2022.

[70] M. Kubat, S. Matwin, *et al.*, "Addressing the curse of imbalanced training sets: one-sided selection," in *Icml*, vol. 97, p. 179, Citeseer, 1997.

[71] A. Fernández, S. García, M. Galar, R. C. Prati, B. Krawczyk, and F. Herrera, *Learning from imbalanced data sets*, vol. 10. Springer, 2018.

[72] E. Y. Pavlenko, I. Eremenko, and A. Fatin, "Computer network clustering methods in cybersecurity problems," *Automatic Control and Computer Sciences*, vol. 56, no. 8, pp. 957–963, 2022.

[73] D. A. Reynolds *et al.*, "Gaussian mixture models.," *Encyclopedia of biometrics*, vol. 741, no. 659-663, 2009.

[74] G. J. McLachlan, "Finite mixture models," *A wiley-interscience publication*, 2000.

[75] C. Chokwitthaya, Y. Zhu, S. Mukhopadhyay, and A. Jafari, "Applying the gaussian mixture model to generate large synthetic data from a small data set," in *Construction Research Congress 2020*, pp. 1251–1260, American Society of Civil Engineers Reston, VA, 2020.

[76] C. M. Bishop, "Pattern recognition and machine learning," *Springer google schola*, vol. 2, pp. 1122–1128, 2006.

[77] R. Foraita, J. Spallek, and H. Zeeb, "Directed acyclic graphs," in *Handbook of epidemiology*, pp. 1481–1517, Springer, 2014.

[78] J. Pearl, *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Elsevier, 2014.

[79] D. Koller and N. Friedman, *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[80] L. N. Martins, F. B. Gonçalves, and T. P. Galletti, "Generation and analysis of synthetic data via bayesian networks: a robust approach for uncertainty quantification via bayesian paradigm," *arXiv preprint arXiv:2402.17915*, 2024.

[81] D. Heckerman, "A tutorial on learning with bayesian networks," *Learning in graphical models*, pp. 301–354, 1998.

[82] K. T. Baghaei, A. Payandeh, P. Fayyazsanavi, Z. Chen, S. B. Ramezani, and S. Rahimi, "Deep representation learning: Fundamentals, technologies, applications, and open challenges," *IEEE Access*, 2023.

[83] D. P. Kingma, M. Welling, *et al.*, "An introduction to variational autoencoders," *Foundations and Trends® in Machine Learning*, vol. 12, no. 4, pp. 307–392, 2019.

[84] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in *International conference on machine learning*, pp. 2256–2265, PMLR, 2015.

[85] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in neural information processing systems*, vol. 33, pp. 6840–6851, 2020.

[86] A. Kotelnikov, D. Baranchuk, I. Rubachev, and A. Babenko, "Tabddpm: Modelling tabular data with diffusion models," in *International Conference on Machine Learning*, pp. 17564–17579, PMLR, 2023.

[87] H. Y. J. Kang, E. Batbaatar, D.-W. Choi, K. S. Choi, M. Ko, and K. S. Ryu, "Synthetic tabular data based on generative adversarial networks in health care: Generation and validation using the divide-and-conquer strategy," *JMIR Medical Informatics*, vol. 11, p. e47859, 2023.

[88] J. C. Rodriguez, "Measuring financial contagion: A copula approach," *Journal of empirical finance*, vol. 14, no. 3, pp. 401–423, 2007.

[89] A. H. Z. Nik, M. A. Riegler, P. Halvorsen, and A. M. Storås, "Generation of synthetic tabular healthcare data using generative adversarial networks," in *International Conference on Multimedia Modeling*, pp. 434–446, Springer, 2023.

[90] Y. Zhang, N. Zaidi, J. Zhou, and G. Li, "Ganblr++: incorporating capacity to generate numeric attributes and leveraging unrestricted bayesian networks," in *Proceedings of the 2022 SIAM International Conference on Data Mining (SDM)*, pp. 298–306, SIAM, 2022.

[91] N. Bouguila and D. Ziou, "High-dimensional unsupervised selection and estimation of a finite generalized dirichlet mixture model based on minimum message length," *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 10, pp. 1716–1731, 2007.

[92] A. Alshantti, D. Varagnolo, A. Rasheed, A. Rahmati, and F. Westad, "Castgan: Cascaded generative adversarial network for realistic tabular data synthesis," *IEEE Access*, 2024.

[93] L. Xu and K. Veeramachaneni, "Synthesizing tabular data using generative adversarial networks," *arXiv preprint arXiv:1811.11264*, 2018.

[94] C. E. Shannon, "A mathematical theory of communication," *The Bell system technical journal*, vol. 27, no. 3, pp. 379–423, 1948.

[95] T. M. Cover, *Elements of information theory*. John Wiley & Sons, 1999.

[96] W. McGill, "Multivariate information transmission," *Transactions of the IRE Professional Group on Information Theory*, vol. 4, no. 4, pp. 93–111, 1954.

[97] S. Manzoor, M. K. Siddiqui, and S. Ahmad, "On entropy measures of molecular graphs using topological indices," *Arabian Journal of Chemistry*, vol. 13, no. 8, pp. 6285–6298, 2020.

[98] K. Church and P. Hanks, "Word association norms, mutual information, and lexicography," *Computational linguistics*, vol. 16, no. 1, pp. 22–29, 1990.

[99] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of machine learning research*, vol. 3, no. Mar, pp. 1157–1182, 2003.

[100] L. Breiman, "Random forests," *Machine learning*, vol. 45, pp. 5–32, 2001.

[101] D. Stiawan, M. Y. B. Idris, A. M. Bamhdi, R. Budiarto, *et al.*, "Cicids-2017 dataset feature analysis with information gain for anomaly detection," *IEEE Access*, vol. 8, pp. 132911–132921, 2020.

[102] R. Mahto, S. U. Ahmed, R. u. Rahman, R. M. Aziz, P. Roy, S. Mallik, A. Li, and M. A. Shah, "A novel and innovative cancer classification framework through a consecutive utilization of hybrid feature selection," *BMC bioinformatics*, vol. 24, no. 1, p. 479, 2023.

[103] X. Zhang, X.-M. Zhao, K. He, L. Lu, Y. Cao, J. Liu, J.-K. Hao, Z.-P. Liu, and L. Chen, "Inferring gene regulatory networks from gene expression data by path consistency algorithm based on conditional mutual information," *Bioinformatics*, vol. 28, no. 1, pp. 98–104, 2012.

[104] J. Hua, W. D. Tembe, and E. R. Dougherty, "Performance of feature-selection methods in the classification of high-dimension data," *Pattern Recognition*, vol. 42, no. 3, pp. 409–424, 2009.

[105] H. Peng, F. Long, and C. Ding, "Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 27, no. 8, pp. 1226–1238, 2005.

[106] M. Wolf, J. Tritscher, D. Landes, A. Hotho, and D. Schlör, "Benchmarking of synthetic network data: Reviewing challenges and approaches," *Computers & Security*, p. 103993, 2024.

[107] A. Kiran and S. S. Kumar, "A methodology and an empirical analysis to determine the most suitable synthetic data generator," *IEEE Access*, 2024.

[108] J. Xu, L. Wei, W. Wu, A. Wang, Y. Zhang, and F. Zhou, "Privacy-preserving data integrity verification by using lightweight streaming authenticated data structures for healthcare cyber–physical system," *Future Generation Computer Systems*, vol. 108, pp. 1287–1296, 2020.

[109] 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," *The American Statistician*, vol. 60, no. 3, pp. 224–232, 2006.

[110] M. Pereira, M. Kshirsagar, S. Mukherjee, R. Dodhia, J. Lavista Ferres, and R. de Sousa, "Assessment of differentially private synthetic data for utility and fairness in end-to-end machine learning pipelines for tabular data," *Plos one*, vol. 19, no. 2, p. e0297271, 2024.

[111] J. Snoke, G. M. Raab, B. Nowok, C. Dibben, and A. Slavkovic, "General and specific utility measures for synthetic data," *Journal of the Royal Statistical Society Series A: Statistics in Society*, vol. 181, no. 3, pp. 663–688, 2018.

[112] P. Thölke, Y.-J. Mantilla-Ramos, H. Abdelhedi, C. Maschke, A. Dehgan, Y. Harel, A. Kemtur, L. M. Berrada, M. Sahraoui, T. Young, *et al.*, "Class imbalance should not throw you off balance: Choosing the right classifiers and performance metrics for brain decoding with imbalanced data," *NeuroImage*, vol. 277, p. 120253, 2023.

[113] E. R. Fernandes and A. C. de Carvalho, "Evolutionary inversion of class distribution in overlapping areas for multi-class imbalanced learning," *Information Sciences*, vol. 494, pp. 141–154, 2019.

[114] T. Lee, K. B. Lee, and C. O. Kim, "Performance of machine learning algorithms for class-imbalanced process fault detection problems," *IEEE Transactions on Semiconductor Manufacturing*, vol. 29, no. 4, pp. 436–445, 2016.

[115] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *2009 IEEE symposium on computational intelligence for security and defense applications*, pp. 1–6, Ieee, 2009.

[116] G. Wei, W. Mu, Y. Song, and J. Dou, "An improved and random synthetic minority oversampling technique for imbalanced data," *Knowledge-based systems*, vol. 248, p. 108839, 2022.

[117] T. Jo and N. Japkowicz, "Class imbalances versus small disjuncts," *ACM Sigkdd Explorations Newsletter*, vol. 6, no. 1, pp. 40–49, 2004.

[118] A. G. Wilson and P. Izmailov, "Bayesian deep learning and a probabilistic perspective of generalization," *Advances in neural information processing systems*, vol. 33, pp. 4697–4708, 2020.

[119] Y. Zhu, Z. Zhao, R. Birke, and L. Y. Chen, "Permutation-invariant tabular data synthesis," in *2022 IEEE International Conference on Big Data (Big Data)*, pp. 5855–5864, IEEE, 2022.

[120] Y. Peng, G. Kou, Y. Shi, and Z. Chen, "A descriptive framework for the field of data mining and knowledge discovery," *International Journal of Information Technology & Decision Making*, vol. 7, no. 04, pp. 639–682, 2008.

[121] X.-L. Meng and D. B. Rubin, "Maximum likelihood estimation via the ecm algorithm: A general framework," *Biometrika*, vol. 80, no. 2, pp. 267–278, 1993.

[122] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[123] M. Scutari, "Learning bayesian networks with the bnlearn r package," *arXiv preprint arXiv:0908.3817*, 2009.

[124] M. Scutari, C. Vitolo, and A. Tucker, "Learning bayesian networks from big data with greedy search: computational complexity and efficient implementation," *Statistics and Computing*, vol. 29, pp. 1095–1108, 2019.

[125] Z. Ge, "Process data analytics via probabilistic latent variable models: A tutorial review," *Industrial & Engineering Chemistry Research*, vol. 57, no. 38, pp. 12646–12661, 2018.

[126] Q. Zhu, J. Su, W. Bi, X. Liu, X. Ma, X. Li, and D. Wu, "A batch normalized inference network keeps the kl vanishing away," *arXiv preprint arXiv:2004.12585*, 2020.

[127] B. A. Alabsi, M. Anbar, and S. D. A. Rihan, "Conditional tabular generative adversarial based intrusion detection system for detecting ddos and dos attacks on the internet of things networks," *Sensors*, vol. 23, no. 12, p. 5644, 2023.

[128] M. Hernadez, G. Epelde, A. Alberdi, R. Cilla, and D. Rankin, "Synthetic tabular data evaluation in the health domain covering resemblance, utility, and privacy dimensions," *Methods of information in medicine*, vol. 62, no. S 01, pp. e19–e38, 2023.

[129] E. W. Frees and E. A. Valdez, "Understanding relationships using copulas," *North American actuarial journal*, vol. 2, no. 1, pp. 1–25, 1998.